YAŞAR UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

COMP4920

Senior Design Project II, Spring 2020

Advisor: *Ahmet Koltuksuz, Ph.D., Assoc. Prof.*

BiometriX

Artificial Intelligence Assisted Biometric Signature on Block Chain

Final Report

5.05.2020

Prepared by:

Taylan AKBAŞ

15070001032

This page intentionally left blank.

PLAGIARISM STATEMENT

This report was written by the group members and in our own words, except for quotations from published and unpublished sources which are indicated and acknowledged as such. We are conscious that the incorporation of material from other works or a paraphrase of such material without acknowledgement will be treated as plagiarism according to the University Regulations. The source of any picture, graph, map or other illustration is also indicated, as is the source, published or unpublished, of any material not resulting from our experimentation, observation or specimen collecting.

Project Group Members:

Name, Lastname	Student Number	Signature	Date
Taylan Akbaş	15070001032		05.05.2020
Ahmet Şen	16070001013		
Doruk Mestanoğlu	16070001011		

Project Supervisor:

Name, Lastname	Department	Signature	Date
Ahmet Hasan KOLTUKSUZ	Computer Engineering		

ACKNOWLEDGEMENTS

We would like to thank our advisor Ahmet Koltuksuz for the his amazing ideas about project and his very precious effort. We are planning a long path for making real to our ideas and we are deeply pleased working with him.

ABSTRACT

The importance of user authentication is a very important aspect especially when the number of users plus the size of the data to be checked against forgery has been increasing exponentially. Biometric Authentication can be done in many ways such as retina, voice, palm, or finger-print recognition. Along with those, the behavioral biometric verification can be used very effectively. A biometric signature is a behavioral biometric recognition that can be done by your actual handwriting signature on -say- a PDA using a digital pen. Biometric recognition systems have produced mostly for the reasons for Identification and Verification. Developing those systems for monitoring and providing access control is now a common practice in a variety of companies, banks, hospitals as well as in public organizations. We intend to develop a mobile application that provides signature verification and transaction management for documents that are hand-signed by clients. Since we deal with the biometric data of the users of the aforementioned organizations, surely the most important issue is security.

ÖZET

Kimlik doğrulama, veri boyutu ve kullanıcı sayısı arttıkça giderek daha önemli hale gelmektedir. Kimlik doğrulama bir çok şekilde yapılabilir. Bu yöntemler arasında retina, parmak izi gibi kişinin biyometrik özelliklerini kullanarak doğrulama sistemlerin yanında davranışsal doğrulama yöntemleri son derece efektif olarak kullanılmaktadır. Biyometrik imza, kullanıcın imzasını dijital bir kalem vasıtasıyla tabletin ekranına atma suretiyle oluşturulan bir davranışsal biyometrik tanımadır. Amacımız, Biometrix mobil uygulamasıyla kullanıcılara istedikleri bir dosyaya biyometrik imzalarını atmalarını, bu imzanın doğrulanması ve her bir imza işleminin yönetimi ve bakımını kolaylıkla gerçekleştirebilecekleri bir ortam sağlamaktır. Sistemin en çok önem vereceği konu saklanacak bilginin hassas olması sebebiyle güvenliktir. Bu ortamda güvenliği sağlarken uluslararası standartlara uygun ve son teknoloji yöntemleri kullanmak son derece kritik olacaktır.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. Introduction

1.1 Description of problem

There are several ways for authentication. People are using passwords, their finger-prints and nowadays retina recognition for authentication. We can divide this methods into two; biometric and standard authentication. Problem of standard authentications are remembering passwords for each system and keeping safe them. The problem of biometric methods is to unalterable of biometric data. If biometric data of users (finger-print, retina) is stolen there is no way to change them. Biometrix also offers to solve very essential problem which is paper consumption.

1.2 Project Goals

In this point Biometrix focusses on solve to all problems that defined above. Everyone remembers own signature, and changes it whenever they want. Because of nature of signature it is quite difficult to mimic and with using Biometrix by companies, governments and all other organizations we may save thousands of tree every year.

1.3 Project Outputs

These are all outputs which we produce for COMP4910

1. Project Assignment Form
2. Requirement Specification Document Revision 1.0
3. Requirement Specification Document Revision 2.0
4. Design Specification Document Revision 1.0
5. Web Poster
6. Final Report

These are predicted outputs which we will produce for COMP4920

7. Biometrix Mobile Application for clients
8. Biometrix Web Application for system administrators
9. Biometrix Data API
10. Final revisions for RSD, DSD,PM and Final Report
11. Product Web Site
12. Poster
13. Slide Presentation
14. Project Summary Description
15. Product Demo Video

1.4. Project Activities and Schedule

1.4.1 Schedule & Activities for COMP4910

Title	Start Time	End Time
Initiation	10/08/2019	11/11/2019
Project Definition	10/08/2019	10/11/2019
Requirement Analysis	10/11/2019	10/26/2019
Research	10/11/2019	11/11/2019
Delivirables	11/01/2019	12/28/2019
RSD Revision 1.0	11/01/2019	11/08/2019
RSD Revision 2.0	12/01/2019	12/19/2019
DSD Revision 1.0	12/10/2019	12/19/2019
Final Report	12/19/2019	12/28/2019
Web Poster	12/21/2019	12/28/2019

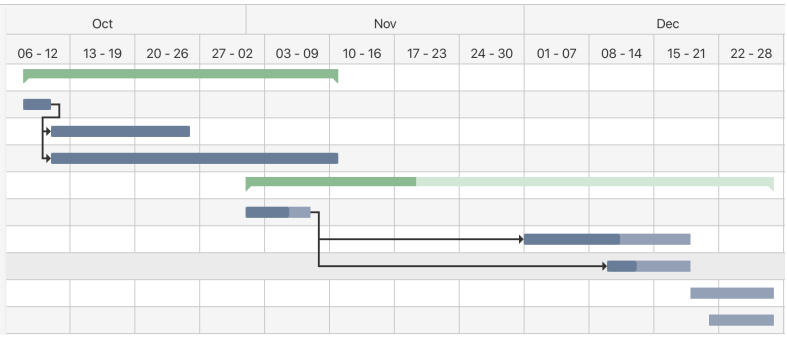


Table 1.

Schedule for COMP4910

1.4.2 Schedule & Activities for COMP4920

Title	Start Time	End Time
Development	02/01/2020	05/04/2020
Web	05/01/2020	05/04/2020
Server	03/01/2020	05/01/2020
Mobil	02/01/2020	04/23/2020
Design	01/14/2020	02/20/2020
UI-UX design	01/14/2020	02/01/2020
Data	02/01/2020	02/20/2020
Architecture	01/24/2020	02/06/2020
Completion	03/10/2020	05/07/2020
Materials	05/04/2020	05/07/2020
Web Site	05/04/2020	05/05/2020
Presentation	05/06/2020	05/07/2020
Poster	05/06/2020	05/07/2020
Delivirables	03/10/2020	05/05/2020
Product Manual	04/09/2020	04/14/2020
Final Report	05/01/2020	05/05/2020
DSD 2.0	03/10/2020	03/16/2020

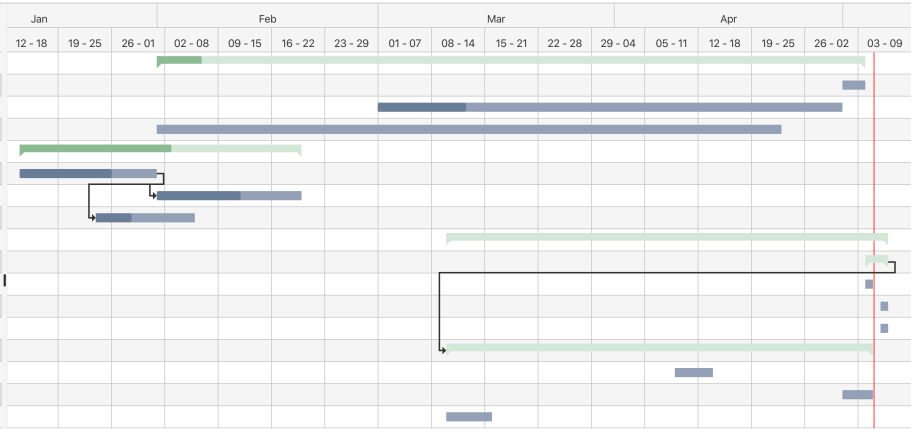


Table 2.

Schedule for COMP4920

2. DESIGN

2.1 High Level Design

Biometrix system will mainly consist of 5 different systems. Biometrix has a database structure for general information that needs to be stored in the system, a blockchain structure to keep critical data safe, a service computer to run artificial intelligence algorithms, provide access to file storage and manage blockchain, structure. Lastly mobile and web applications for end-users and system administrators to provide user-interface.
Detailed high level designed information can be found in Appendix A: Design Specification Document - High-Level Design.

3. IMPLEMENTATION, TESTS and TEST DISCUSSIONS

3.1. Implementation of the Product

Implementation of the Biometrix environment divided three different components. Since the main objective is signature verification, we implement a mobile application to capture the signature first.

3.1.1. Implementation of Biometrix Mobile Application

There are a small number of a library to capture signature data for mobile platforms. We use T1 Autograph Library for capturing biometric data from the signer. Library provides 10 features for each point in a signature. The number of points changes for each signature.
SignatureObject described the following table.

Property	Type	Description
imageData	NSData	Raw image data of the non-clipped signature.
imageView	UIImageView	A retina-ready view of signature.
pdfData	NSData	Raw pdf data of the non-clipped signature.

pdf	DocumentRef	Signature in pdf format (retained only by T1Signature).
svgString	NSString	SVG format suitable for web use.
xmlString	NSString	ISO/IEC 19794-7 XML formatted biometric data string.
hashString	NSString	Hash string (user-defined or auto-generated).
frame	CGRect	Location and size of signature within the enclosing view.
timestamp	NSTime Interval	Timestamp of signature referenced to system uptime.
rawPoints	NSArray	Array of strokes. Each stroke is an NSArray containing T1SignaturePoint objects describing the signature.

Table 3.

Signature data

SignaturePoint objects are available from the raw points property of T1Signature. They contain raw data as captured from the digitizer.

Property	Type	Description
Location	CGPoint	Position relative to enclosing view, in points.

Velocity	CGPoint	Instantaneous velocity of drawing tool, in m/s.
Acceleration	CGPoint	Instantaneous acceleration of drawing tool, in m/s^2.
Timestamp	NSTimeInterval	Timestamp of point referenced to system uptime.
Azimuth Angle	float	Angle of the perpendicular projection of the pen onto the writing plane. In radians, with zero radians pointing along the positive X axis.
Altitude Angle	float	Angle of pen tilt from perfectly vertical (PI/2) to laying flat (0). In radians.
Pressure	float	Force of drawing tool at this point. Range is 0-1.0f.
Diameter	float	Diameter of stroke at this point as rendered by the framework. Units are in points.
ID	NSUInteger	Unique point identifier.

Table 4.

Signature point

The following table demonstrates the first three rows of sample data that collected from a client's signature:

Attribute	1	2	3
$x(t)$	162.9163818359	194.6800537109	219.0797119140
$y(t)$	157.3918457031	157.5191040039	158.9825744628
$p(t)$	0.33494857	0.32474792	0.32963637
$v_x(t)$	0.08181615447	-0.15015119140	-4.09824560038
$v_y(t)$	0.178507973403	0.122244840739	0.582242803698
$a_x(t)$	0.253897840726	0.251495421672	0.181825246496
$a_y(t)$	-0.00094832360	0.001007593836	0.010905721495
altitude	1.0273546	1.0225481	1.0196947
azimuth	1.7819322	1.8223222	1.8438524
timestamp	1039511.052	1039511.069	1039511.086

Table 5.

Sample collected data

Biometrix Mobile App collects five signature from clients for creating a model.

The same operation occurs in signature verification but this time the app collects just one signature, to compare the client's model. All details about model and signature operations explained in the Implementation of Biometrix Data API.

In the verification phase, the mobile app provides more information about the transaction in contrast to registration. A transaction information content consists of:

- Transaction ID
- Client ID
- End-user ID
- Document ID
- Timestamp
- Score

- Threshold
- Difference
- Transaction Result
- Latitude of Device
- Longitude of Device
- Signature Raw Data
- Signature Image
- Signer Image

Biometrix Mobile App also uses the Firestore Database for access transaction details, user credentials, client information, and documents. We implement all features described in RSD Final Revision and implement according to viewpoints that described in DSD Final Revision.

3.1.2. Implementation of Biometrix Data-API

The most important component of the Biometrix environment is Data-API. Signature model generation, signature verification, writing the model on blockchain operations runs in the API.

We spend most of our time to choose a robust algorithm for signature recognition. There are many approaches to data classification of signature. According to previous works we decided to use Dynamic Time Warping to calculate the distance between signatures. This approach shows that the system gives quite good results.

3.1.2.1 Signature Verification

I. Preprocessing

In the preprocessing phase, we normalized location of the signature and removed the jagged of signature. When we capture a signature, the signer could start to sign any location of the surface thus we normalized x and y locations. The tablet device may have a low resolution. So smoothing signature is an important preprocessing operation before extracting local features.

II. Feature Extraction

Since signature data already collected from a mobile device as mentioned in Table 3, we have to extract local features.

After extraction, we will have 20 features for each point using for signature recognition. All features defined in the following table.

Feature name	Description
$x(t)$	The normalized x coordinate
$y(t)$	The normalized y coordinate
$p(t)$	The pressure
altitude(t)	The altitude angle
azimuth(t)	The azimuth angle
$v_x(t)$	Speed in x direction
$v_y(t)$	Speed in y direction
$a_x(t)$	Acceleration in x direction
$a_y(t)$	Acceleration in y direction
$a(t)$	Absolute acceleration
$a_t(t)$	Tangential acceleration
$p'(t)$	Press derivation
$\alpha(t)$	The α angle between the absolute velocity vector and the x axis
$sin\alpha(t)$	Sine of the α
$cos\alpha(t)$	Cos of the α
$\alpha'(t)$	Derivation of α angle
$sin\alpha'(t)$	Sine of the $\alpha'(t)$

$cos\alpha'(t)$	Cos of the $\alpha'(t)$
$\beta(t)$	The angle between two adjacent line segments at each coordinate

Table 6.

Extracted features

After extraction, the signature can be considered as a feature matrix:

$$O = \{o_{k,t}\}_{t=1..T}^{k=1..20}$$

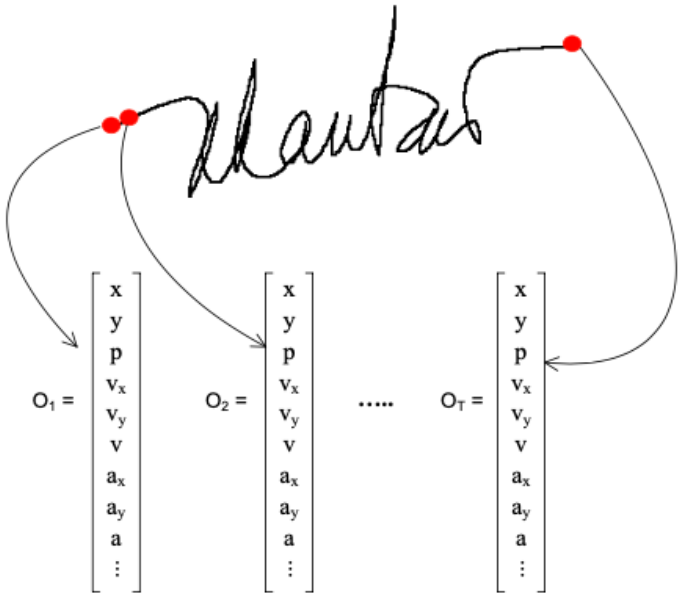


Figure 1.

The signature observation

III. Feature Normalization

Feature scaling through standardization is very important since our values in different ranges. We use Z-score normalization for each point using the following equation:

$$z_i = (x_i - \mu)/\sigma$$

IV. Training

After we have extracted signatures in our database, we can create a signature model and calculate the threshold value. Distance between two points on two different signature can be calculated with Euclidian distance. To compare two signatures with different length, we use Dynamic Time Warping.

DTW Algorithm:

```
int DTWDistance(s: signature1 [1..n], t: signature2 [1..m]) {
   DTW := array [0..n, 0..m]

   for i := 1 to n
      for j := 1 to m
         DTW[i, j] := infinity
   DTW[0, 0] := 0

   for i := 1 to n
      for j := 1 to m
         cost := d(s[i], t[j])
         DTW[i, j] := cost + minimum(DTW[i-1, j  ],    // insertion
                                     DTW[i  , j-1],    // deletion
                                     DTW[i-1, j-1])    // match

   return DTW[n, m]
}
```

where DTW[i, j] is the distance between s[1:i] and t[1:j] with the best alignment.

Similarity Calculation:

To calculate the similarity between two signature we follow these steps:

1. Calculate optimal alignment using DTW
2. Normalize distance of alignment

We calculate similarity for each signature pairs from our model

Threshold Calculation:
Since we have five signature for the model, we have 10 different similarities. To calculate the threshold, we calculate the average value of similarities.

V. Verification

Once we calculate the threshold value, we can ready for the verification phase. To calculate the verification score of test signature, we need to make a similarity calculation between test signature and each signature in the model. Average of similarities gives the score of test. Last decision made with following condition:

IF test score - threshold >= 0.2
Forgery
ELSE
Geniue

In this way, we can recognize if the signature is a forgery or not.

VI. Experiments

To test our verification system, we collected signatures for 5 different signers after create their model. Collected signatures distributed as:

- 10 real signatures
- 10 professional forged signatures
- 10 random forgery signatures (obtained from other users) for each signer.

We calculate Equal Acceptance Rate (%EER) for the system: 7%

3.1.2.2 Adding New Block To Blockchain

Biometrix Data-API demonstrate how we can store our sensitive data on a blockchain. When a new user model created, the API generates a custom structure that consists of model data. Then it creates a new block and adds to our blockchain structure. This application is proof of concept so any decentralization operation has not placed here. The following representation shows how to store each data in our blockchain.

"**data**": {
"**client**": "1",
"**threshold**": 2.0694338110917636,
"**signatures**": {5 items}
},
"**prev_hash**":"83d7411b8069a6f1ffa6aeee6379f967a2d666eb65cc5cf4eeef318b058c5b55",
"**timestamp**": 1586800014.914481,
"**proof**": 95,
"**index**": "2"

3.1.3. Implementation of Biometrix Web-Admin

Biometrix Web App is implemented for monitoring transactions and user operations by System Administrators. This version of the app does not include many backend concerns like login operation and security issues.

We use a ready-to-use template for frontend and redesigned according to our needs. To be able to access our Firestore database we implement a considerable amount of scripts. The main objective is here, separate privileges of users and provide more functionality to the Biometrix environment. Since our sensitive data stored in our server, the Biometrix Web-Admin application has access to our API.

4. CONCLUSIONS

4.1. Summary

The team considered all requirements and design considerations so far. We accomplish our purpose that is developing a prototype in defined constraints at the end of this year.

4.2. Cost Analysis

BIOMETRIX SYSTEM COST PROFILE		
COST CATEGORY	**LIFETIME**	**TOTAL**
Hardware		$ -
iPad Air	$ 629	$ 629
Apple Pencil	$ 99	$ 99
Manpower (1 $ = 7,02 TL) 25 hours per week (8 months) x Team member)	$ 3.200	$ 3.200
TOTAL PROJECTED COSTS	$ 3.928	$ 3.928
CUMULATIVE TOTAL PROJECTED COSTS	$ 3.928	

Table 7.

Cost analysis for Biometrix

4.3. Benefits of the Project

The most substantial benefit of Biometrix is providing a secure and easy to use a system for signing a document. The project will be ready to use all companies/organizations and government offices. Usage of application will reduce paper consumption and it is very crucial for our world. It will help companies go paperless and the benefits of the technology go well beyond just eliminating printer projects.

4.4 Future Work

We are very excited about the future of this project and we believe this prototype will be a real product soon. Almost all companies/organizations and/or government offices need authentication methods and biometric signature is exactly what they are looking for.
Since Biometrix includes many different components, we will improve almost all features. Starting to implement our own signature capturing library, owning a powerful server, and ensure the security of the system would make sense. Also blockchain part of this project has to be realized by the project team.

References

I. Akbas T.(2020). Biometrix : Artificial Intelligence Assisted Biometrix Signature on Block Chain Requirements Specification Document (Final Revision).

II. Akbas T.(2020). Biometrix : Artificial Intelligence Assisted Biometrix Signature on Block Chain Design Specification Document (Final Revision).

III. ISO/IEC 19794-7:2014

IV. Miguel-Hurtado, Oscar. (2011). Online signature verification algorithms and development of signature international standards.

V. Chan, Siew & Tay, Yong Haur. (2006). Online Signature Verification using Dynamic Time Warping.

VI. Podio, Fernando L., and J S. Dunn. "Biometric Authentication Technology: From the Movies to Your Desktop." *NIST*, 17 Feb. 2017.

VII. Sam Krasnik, Ron Hose "Signature Based Authentication", 2015.

VIII. Ravi Das (Writer/Revisions Editor), Author at Infosec Resources." *Infosec Resources*, https://resources.infosecinstitute.com/author/ravidas/.

IX. Patil, B. V., & Patil, P. R. (2018, February). An Efficient DTW Algorithm for Online Signature Verification

X. Toni Giorgino (2009). Computing and Visualizing Dynamic Time Warping Alignments in R: The dtw Package. Journal of Statistical Software, 31(7), 1-24, doi:10.18637/jss.v031.i07.

APPENDICES

APPENDIX A: REQUIREMENTS SPECIFICATION DOCUMENT

Yasar University

COMP4920

Senior Design Project II, Spring 2020

Advisor: *Ahmet Koltuksuz, Ph.D., Assoc. Prof.*

Requirements Specification Document

for

BiometriX

Artificial Intelligence Assisted Biometric Signature on Block Chain

Revision 3.0

Prepared by:

Taylan AKBAŞ

15070001032

This page intentionally left blank.

TABLE OF CONTENTS

Revision History

Revision	Date	Explanation
1.0	08.11.2019	Defined general perspective of Biometrix and its initial requirements.
2.0	28.12.2019	Defined specific requirements, actors and their use cases
3.0	24.04.2020	Finalized the document

1. Introduction

Authentication of the user is enhancing very critical to do accessing data and business transactions. While many different techniques applied for authentication, biometrics authentication has been gaining among the industries that highly incorporate securities in their daily activities.
Behavioral biometric attributes have characteristics such as universality, uniqueness, permanency, collectability, and comparability. The attributes used for authentication include iris, hand geometry, face, and fingerprints, but the requires special hardware to capture.
Signature verification is one of the prior and solid techniques in behavioral biometrics. The most substantial advancement of signature verification conjugated with lower cost and technology advances in tablets and mobile phones.

1.1 Purpose

The purpose of this document is an attempt to provide support information for accurate and secure biometric signature verification systems: *Biometrix* and explain the functionality of the system, requirements, and features it provides.

1.2 Intended Audience

This Requirement Specification document is intended for:

- Developers who can review the project's capabilities and more easily understand where their efforts should target to improve or add more features to it.
- End-users of this application who wish to read about what this project can do.

1.3 Scope

For a long time signature has always been associated with that of law requirement. The usage of a signature confirms the identity of every individual. It creates a legally binding contract or agreement between two or more parties. Biometrix is a mobile application for signing any document in an accustomed way with a state of the art level of security. Principally, Biometrix will provide conformity for companies from all industries, banks, hospitals, government offices, and other organizations. The application will be used for dynamic (online) signature recognition which will be assisted by our AI algorithms. It also can be used for distinguishing to the forgeries for forensics purposes. The most significant benefit of signature recognition is that is highly resistant to impostors.

After the user puts own signature on a mobile device, Biometrix extracts dynamic pieces of information. This information consists of x-y-z coordinates, velocity, acceleration, time difference, pen tip force, azimuth, and elevation angle of the pen and rotation about the pen axis. After extracting other local features it compared with the user client model in advanced statistical methods. So, it is quite difficult to mimic the behavioral patterns which are inherent in the process of signing by the signer.
There is always a concern amongst the users of biometric systems what if their templates will be stolen. Since these templates immutable because of its natural structure (such as iris, retina, or fingerprint), it will not be changed with a new template when the system is compromised. However, with our modality, the actual structure of the signature can be changed very quickly because of its fluid and dynamic nature. Thus, it is very difficult to spoof.
When compared to the other Biometric modalities, signature recognition is deemed to have accepted by the public. Biometrix provides an environment for clients who has numerous clients or/and employee. Clients able to managing all documents and realize signature processes securely.
Lastly, Biometrix aims to solve another vital problem on our planet: tree consumption for paper. With start to using Biometrix by both public and private corporations, we may save billions of trees every year.

1.4 Definitions, Acronyms, and Abbreviations

Terms	Definition
Client	Clients
LEU	Limited End User
SA	System Administrator

1.5 References

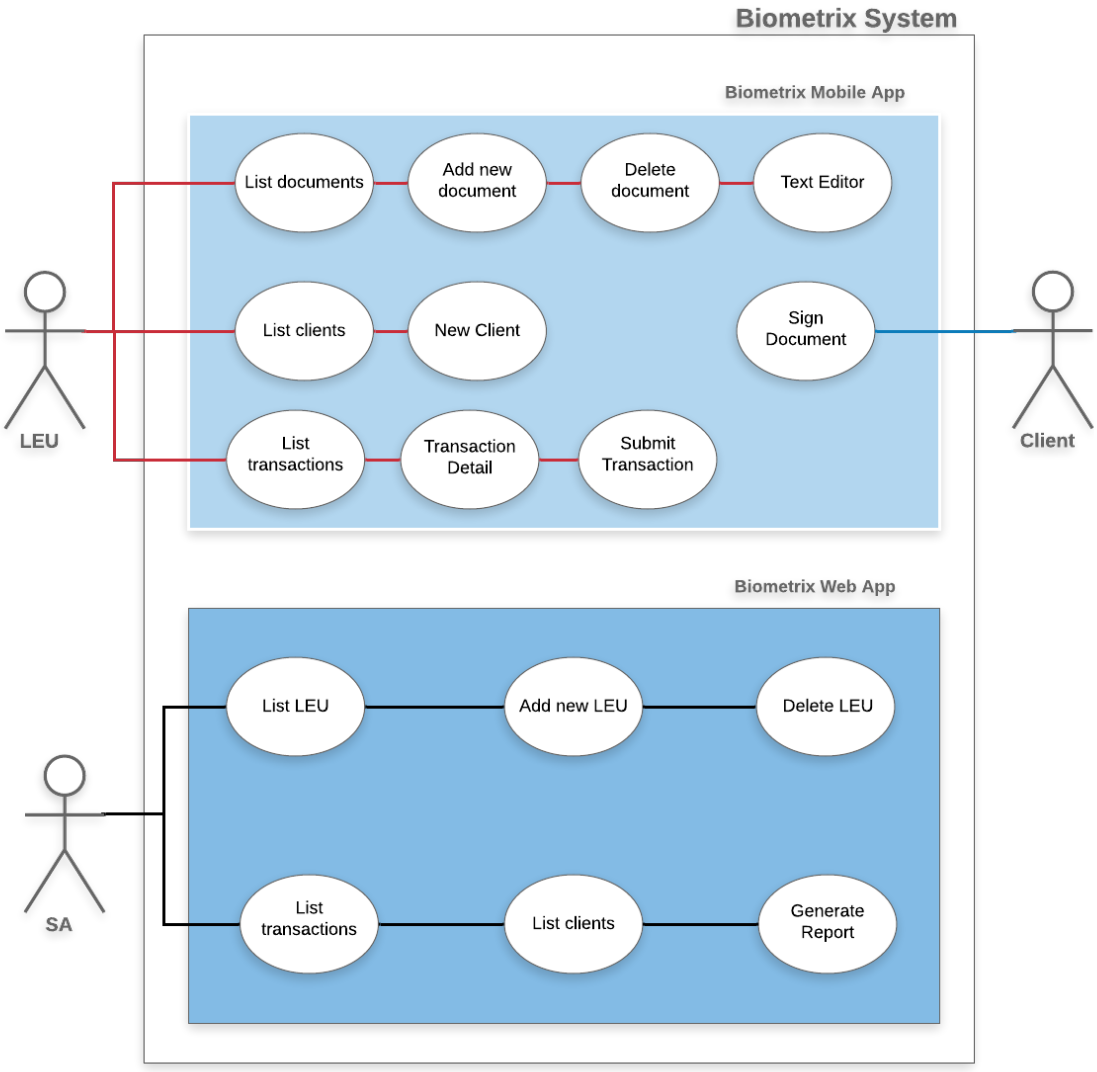
I. IEEE Std. 830-1984 (1993,1998)

II. ISO/IEC 19794-7:2014

2. General Description

2.1 Product Perspective

Biometrix will consist of a mobile application that allows signing any document with its hand-written signature on a digital tablet for limited end-users, and management of subsystem for administrative end users. It also will include an online web application that allows maintaining the entire system for system administrators. It is a distributed software system and it can be customizable for any public-private enterprises. Once registered to the system as an administrative end user, a user can manage user, monitor transactions, and all user information in own subsystem. When registered as a Limited End User, the real transaction occurs. Biometrix gives users the ability to adding documents and signing them. **Figure 1.** shows all of the actors and their functions that will be implemented in the system environment.



2.2 Product Features

2.2.1 Limited End-User Features

- Document management
- Registering new client
- Adding a new signer feature allows creating new signer accounts in the system.
- Transaction monitoring feature will allow to monitoring the latest transactions

2.2.2 System Administrator Features

- The dashboard view will allow the user to monitor current condition of the system. It will provide information about the status of transactions and client statistics.
- System interface will provide an internal interface to monitor error logs, user logs, and detailed information about these logs. Also, an interface to control subsystems and able to generate reports contain information about transactions.
- Managing Profiles will help to control both limited end-users and clients' information.

2.2.3 Client Features

- The signing process includes four principal steps; specifying the file (chosen by LEU), getting the result, and making a transaction.

2.3 User Classes and Characteristic

2.3.1 System Administrators: User class with domain system privileges which allows maintaining the entire system, manage enrollment, maintain all devices, users, and gives ruling over an ability on own subsystem.

2.3.3 Limited End User: User class with limited class privileges, which allows them to operate the signing process and document management.

2.3.3 Client: User class with limited class privileges, which allows them to sign a document.

3. Specific Requirements

3.1 Summary of Requirements

This section outlines the use cases for each of the actors and detailed information about requirements.

The system consists of three major roles; system admin, limited end-user, and client. For the purpose of managing the subsystem, the system admin will have a management interface. The management interface is an interactive interface that includes device management, transaction monitor, and user profile functions with necessary backend implementation. Limited users will have an interface that will include the functions of signature enrollment and its verification and file management.

3.1.1 Hardware requirements

The subsystem requires two main hardware components. Firstly, to capture a signature, a tablet with a high touch-sensitive screen. Secondly, to sign, a pen which is compatibly selected for the tablet. The entire system will have a dedicated server and computers which will be connected to the server.

3.1.2 Software requirements

Both end-users will have a common mobile application installed on their tablets for their actions. The system administrator will have a web application to maintain databases and subsystems.

3.2 Requirement List

#	Requirement	Use Case
1	To sign any document chosen by LEU expects identification of signer and verification of signature.	Sign a document
2	To display workspace of sub-system	List documents
3	To creating a new document with document context using by text-editor	Add new document
4	To delete document from workspace choose by LEU	Delete document
6	To list all signer accounts in the system	List clients

#	Requirement	Use Case
5	To create a new client, names, signature and id details are must be declared. To enroll new signature, system needs 5 different sample taken by client.	Register new client
7	To list all transactions and its detailed information	Monitor transactions
8	To list all LEUs in the sub-system	List LEUs
9	To add new LEU account identity information	Add new LEU
10	To activate/deactivate a LEU in the sub-system	Delete LEU
11	To generate a report. This report will include transferred file's hash , signer's details, timestamps.	Generate report

3.3 Actors and Use Cases

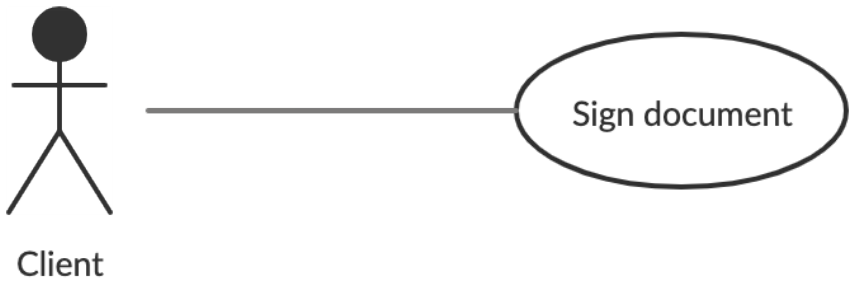
3.3.1 Actors

Actor	Description
Client	Any client of sub-modules. Each client has identity information and a signature model consists of signatures. All actions of the signer depend on the permissions of LEU.
LEU	Each LEU member of the sub-module is responsible for all stages of the signing process. The system expects two main necessities; choosing a document and getting a signature from the signer. Signer person will be changed according to requirements of sub-modules
SA	System Administrator is the management position of the entire system. SA is responsible for monitoring systems, AEU management, systems backup, and generating reports for any critical problem as an option.

3.3.2 Use Cases

3.3.2.1 Sign a document

Overview	
Title	Sign a document
Actor	Client
Description	Signature capturing and verification process occurs this case. Conformable to result of signature verification, transaction ends up.
Precondition	LEU must choose document from workspace and client must authenticated. Authentication can be done with sensitive information such as signer's identity number or/and signature verification.
Postcondition	System generates a log information about transaction

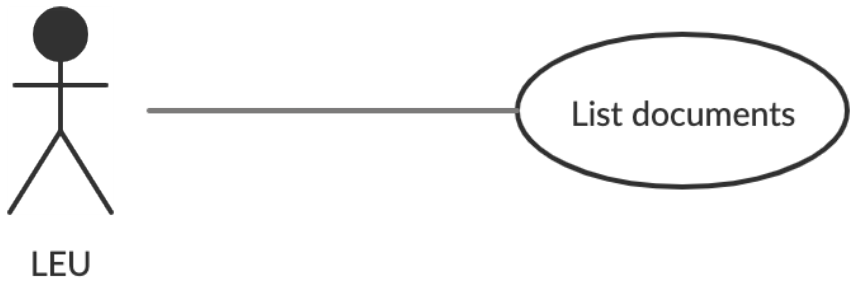


Basic Flow

#	Actor Action	System Response
1	None	Lists all documents that created before and ready to sign.
2	The actor selects one document.	Displays a login form for getting signer's credentials.
3	Signer signs the document	Presents a message according to result of verification of signature

3.3.2.2 List documents

Overview	
Title	List document
Actor	LEU
Description	Lists document workspace

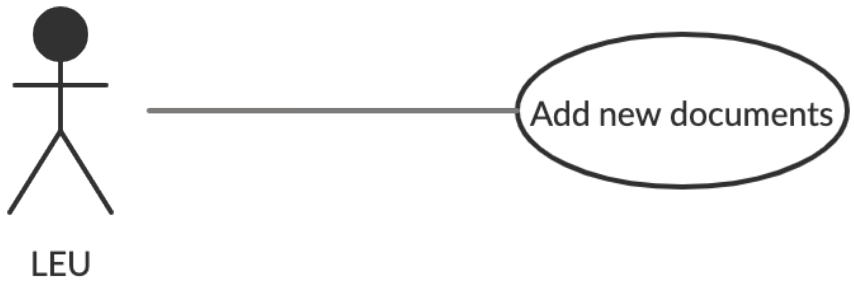


Basic Flow

#	Actor Action	System Response
1	None	Displays documents
2	The actor selects a document	Loads document content to editor

3.3.2.3 Add new document

Overview	
Title	Add new document
Actor	LEU
Description	User can create a new document for signing and uploads to workspace.

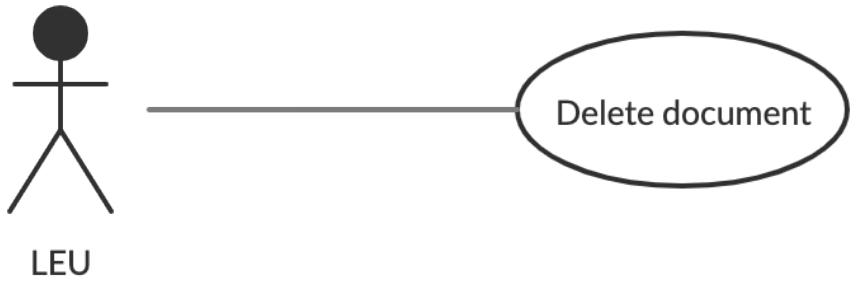


Basic Flow

#	Actor Action	System Response
1	None	Displays simple a text editor
2	The actor writes context of document	None
3	Save the document	Response message

3.3.2.4 Delete document

Overview	
Title	Delete document
Actor	LEU
Description	User can delete document



Basic Flow

#	Actor Action	System Response
1	None	Displays simple a text editor
2	The actor selects name of document	Loads the content of document to editor.
3	Delete the document	Response message

3.3.2.5 List clients

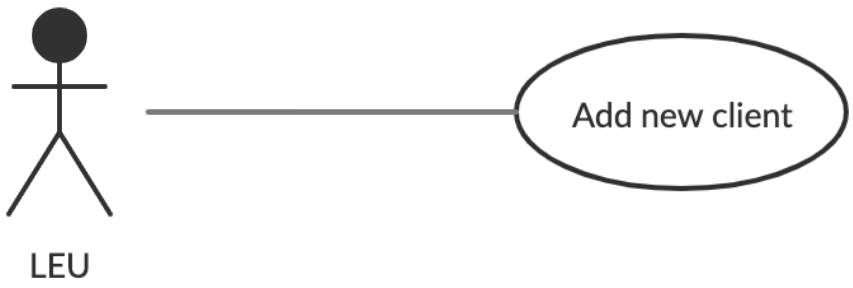
Overview	
Title	List clients
Actor	LEU & SA
Description	Registered signers are kept in the system for several purposes. LEU and SA could monitor all the signers registered in the system.

Basic Flow

#	Actor Action	System Response
1	None	Lists all signer information and their transactions in detailed.

3.3.2.6 Add new client

Overview	
Title	Add new signer
Actor	LEU
Description	When LEU needs to create a new signer personal information are recorded. Other signer information details specified by LEU separately. After that, signature enrollment process starts. A frame pops up for signing. It continues until reached 5 try. Lastly, Biometrix creates a signature model for each client.
Precondition	Signer must not be registered with same ID.
Postcondition	Before adding the new signer in sub-system, signature of the signer must be verified.

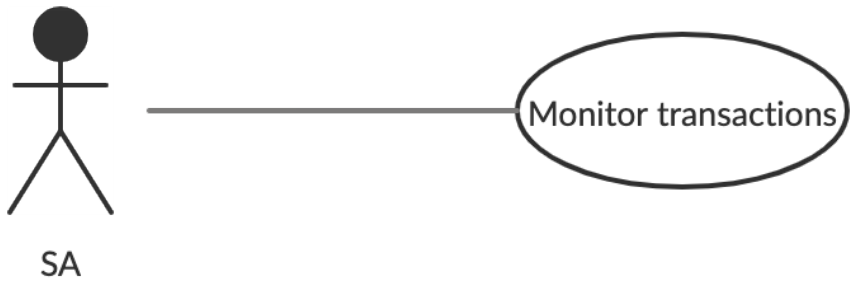


Basic Flow

#	Actor Action	System Response
1	None	Displays a form for taking signer identity information
2	Confirm identity of signer	Displays signing field.
3	Signer signs to the field	Presents a message confirming to new signer informations

3.3.2.7 Monitor transactions

Overview	
Title	Monitor transactions
Actor	SA
Description	The app logs every transaction with timestamp, name of file, sender, target receiver etc. for security purposes. In case of an unexpected situation, SA could monitor these logs and gather detailed information about all the file transfers occurred.
Postcondition	System generates a log information about transaction

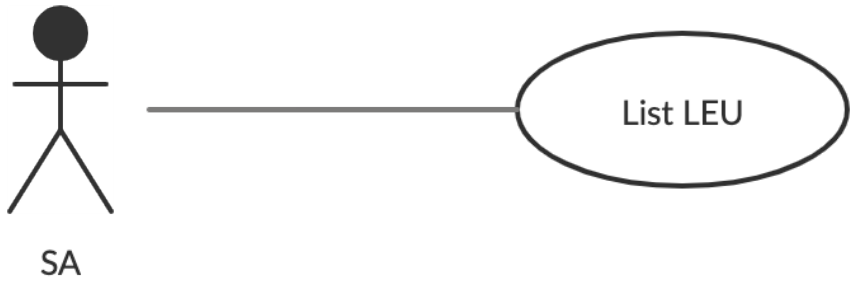


Basic Flow

#	Actor Action	System Response
1	None	Lists all transactions

3.3.2.7 List LEU

Overview	
Title	List LEU
Actor	SA
Description	SA is responsible for LEU management. Therefore, SA could list LEUs registered to the system and monitor detailed information about all LEUs.
Precondition	None
Postcondition	None

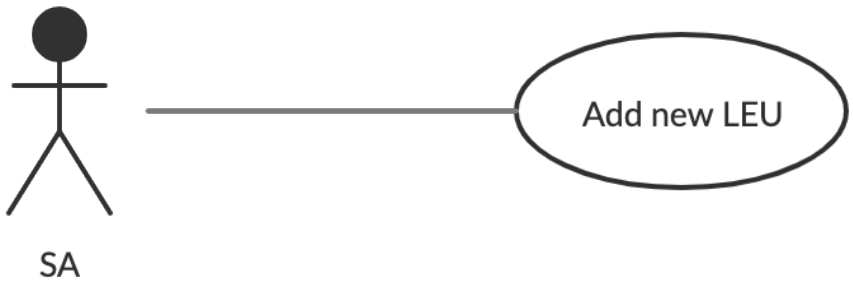


Basic Flow

#	Actor Action	System Response
1	None	Lists all LEU informations in detailed.

3.3.2.8 Add new LEU

Overview	
Title	Add new LEU
Actor	SA
Description	When a new staff is joins to the company, his or her identity information, timestamp and staff number are recorded as well as signature. System logs new LEU entry. Then, LEU could start using the system.
Precondition	New LEU must not be registered before.
Postcondition	None

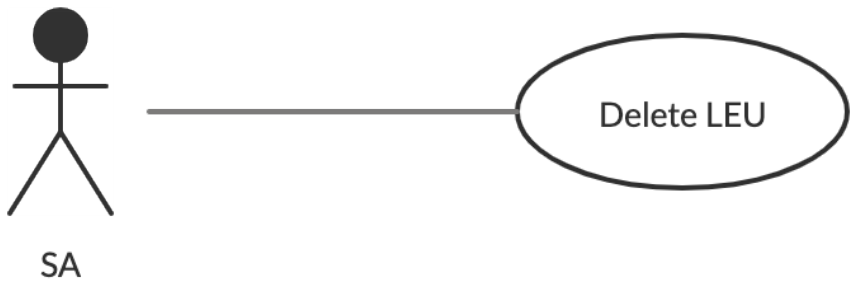


Basic Flow

#	Actor Action	System Response
1	None	Displays a form for taking LEU's identity information
2	Registers the new LEU	Presents a message confirming to new LEU informations

3.3.2.9 Delete LEU

Overview	
Title	Delete LEU
Actor	SA
Description	Old staff should be unregistered from the system. SA could block LEU accounts with a proper reason. LEU accounts are never deleted permanently. This prevents data loss. System logs reason, subject LEU account and timestamp.

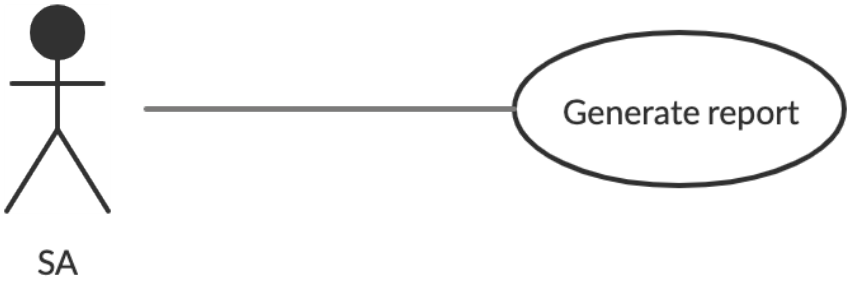


Basic Flow

#	Actor Action	System Response
1	None	Lists all .sd documents which created before and ready to sign.
2	The actor selects one document.	Displays a login form for getting signer's credentials.
3	Signer signs the document	Presents a message according to result of verification of signature

3.3.2.10 Generate Report

Overview	
Title	Monitoring entire system
Actor	SA
Description	When any needs occurs in forensics area, system generates a report which includes timestamps, signer's details, file's hash etc.
Precondition	Permissions* (It will be explained in further versions of this document)
Postcondition	None



Basic Flow

#	Actor Action	System Response
1	Selects a transaction and request to download transaction report	It generates a report from specified transaction

APPENDIX B: DESIGN SPECIFICATION DOCUMENT

Yasar University

COMP4920

Senior Design Project II, Spring 2020

Advisor: *Ahmet Koltuksuz, Ph.D., Assoc. Prof.*

High Level Design
Design Specification Document

for

BiometriX

Artificial Intelligence Assisted Biometric Signature on Block Chain

Revision 3.0

Prepared by:

Taylan Akbaş

15070001032

This page intentionally left blank.

TABLE OF CONTENTS

Revision History

Revision	Date	Explanation
1.0	28.11.2019	Defined initial high-level design.
2.0	16.03.2020	Defined several viewpoints, their explanations and diagrams.
2.1	30.04.2020	Defined all subsystems with related context of viewpoints
3.0	04.05.2020	Finalized the document

1. Overview

1.1 Scope

Simply the developed product is a mobile application that can sign a document using a stylus and a web application that can manage administrative operations. Objectives of the Biometrix are:

- Providing an environment for signing processes
- Registering a signature with its model for further operations
- Signature verification
- Monitoring transaction details
- Generation transaction analysis files

This work can be used to commercial, governmental or, military software that runs on tablets. This document is prepared in IEEE 1016-2009 standards.

1.2 Purpose

The purpose of this document is to describe the design of the software product for answering how to design structured and how the process of it develops.

1.3 Intended audience

The Intended audience of software design description is all developers who can review the project's capabilities and more easily understand where their efforts should be targeted to improve or add more features to it and end-users who wish to read about what this project can do.

1.4 References

I. IEEE. IEEE Std 1016-2009 IEEE Standard for Information Technology – System Design – Software Design Descriptions. IEEE Computer Society, 2009.

II. Akbas T.(2020). Biometrix: Artificial Intelligence Assisted Biometrix Signature on Block Chain Requirements Specification Document (Revision 3.0).

III. ISO/IEC 19794-7:2014

IV. Fakhroutdinov, Kirill. "The Unified Modeling Language." UML Diagrams - Overview, Reference, and Examples., www.uml-diagrams.org/.

2. Definitions, Acronyms and Abbreviations

All the definitions, acronyms and abbreviations which are used in this document are described following table.

DSD	Design Specification Document
RSD	Requirement Specification Document
UML	Unified Modeling Language
Class Diagram	A type of static structure diagram in UML that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes
LEU	Limited End-User
SA	System Administrator
Context Diagram	The context diagram is used to establish the context and boundaries of the system to be modeled.
JSON	JavaScript Object Notation
PDF	Portable Document Format
REST	Representational State Transfer

API	Application Interface
SQL	Structured Query Language
UI	User Interface
Transaction	Sign document, signature verification and storing sensitive data on server
Hash	A hash function is any function that can be used to map data of arbitrary size to fixed-size values.
Segue	Proceed to what follows without pause. It used for application views

3. Conceptual Model for Software Design Descriptions

This section establishes a conceptual model for Biometrix. The conceptual model includes system terminology and information for stakeholders who will interest this software.

3.1 Software Design in Context

Biometrix system will mainly consist of 5 different systems. Biometrix has a database structure for general information that needs to be stored in the system, a blockchain structure to keep critical data safe, a service computer to run artificial intelligence algorithms, provide access to file storage and manage blockchain structure. Lastly mobile and web applications for end-users and a web application for system administrators to provide user-interface.

3.2 Software Design Descriptions within the Life Cycle

The main milestones of this system requirement analysis, design description analysis, UI design, implementation, verification, and validation.

3.2.1 Influences on DSD Preparation

The primary influence on the system design process is the Biometrix RSD 2.0 document. We considered all functional and nonfunctional requirements and relations between them.

3.2.2 Influences on Software Life Cycle Products

This document influences the content of other documents of the project.

- *Requirement Specification Document*. Design decisions or constraints discovered during preparation RSD.
- *Product Manual*

3.2.3 Design Verification and Design Role in Validation

Verification is the process that we will test the Biometrix system whether it meets a set of design specifications using RSD and DSD documents. We are responsible for whether all functional and nonfunctional requirements correctly implemented according to the requirements of RSD and DSD documents.

4. Design Description Information Content

4.1 Introduction

Software design description of Biometrix system explains how the system will be designed and implemented. This section contains identification of the DSD, identified design stakeholders, identified design concerns, selected design viewpoints, design views, overlays, rationale, and languages.

4.2 DSD Identification

Biometrix will be released by the first week of May 2020 after validation and verification tests. The prototype of the mobile application will be shown the beginning of April 2020. Modification and distribution can only be done by copyright holders who are members of development team. Scope, references context and summary can be found under the section "Overview". Glossary can be found under the section "Definitions, Acronyms and Abbreviations".

4.3 Design Stakeholders and Their Concerns

Stakeholders include a developer team at Yasar University Department of Computer Engineering. Mainly concerns of the stakeholders are verification performance of signature and making critical information.

4.4 Design Views

Biometrix organized into several design views to help design stakeholders about focusing on design details from a specific perspective. Each design viewpoint concerns defined topics in the previous subsection.

4.5 Design Viewpoints

This section describes context, logical, dependency information, patterns, and interface viewpoints.

Context Viewpoint: It describes the relationships, dependencies, and interactions between the system and its environments such as end-users and system admins.

Logical Viewpoint: It elaborates class structures and the interaction between them. It includes a class diagram that defines objects and classes and relationships between them.
Dependency Viewpoint: It describes relationships of interconnection and access Amon entities. It gives information about shared information and the order of execution of these components.
Information Viewpoint: It describes data items, types and classes, data stores and access mechanisms and also it gives information about data attributes.
Pattern Viewpoint: It describes design patterns and usage of design patterns in the system.
Interface Viewpoint: It provides information for designers, programmers and, testers before proceeding with the detailed design.

4.6 Design Elements

A design element is any item occurring in a design view. It may have some of these subclasses; design entity, relationship, design attribute and, design constraints.

4.7 Design Rationale

Biometrix system structure will consist of mainly five systems. Main database to manage the subsystems details. Child databases manage the subsystems all the information (except critical ones such as signatures ext.). Blockchain used to keep this critical information to make it secure. A mobile application and a web application to make the file transactions, signer registrations, signing document and monitoring all details about the system. For the last one a service computer to make the management the whole structures.

4.8 Design Languages

In this document, UML will be used as the modeling language for the diagrams. Design languages are selected as a part of the design viewpoint specification.

5. Design Viewpoints

5.1 Introduction

This section defines several design viewpoints of Biometrix for DSD. It demonstrates to corresponding design concerns and design languages. Table 1 summarizes these design viewpoints. For each viewpoint its name, design concerns and, design languages listed.

Design viewpoint	Design concerns
Context	System services and users
Logical	Abstraction and reusable components
Dependency	Dividing sub-systems according to their purposes
Information	Data Types
Pattern	Design pattern & Models
Interface	UI Design

Table 1.
Summary of design viewpoints

5.2 Context Viewpoint

The context viewpoint of Biometrix system provides a "black box" perspective and it shows relationships between the system and its environment.

5.2.1 Design Concerns

Design concerns include services, actors, and system boundaries. Biometrix Application separated to mobile and web interface by intended use. Biometrix Mobile is a simple application to be installed on special devices that the company uses. All end-user communication of the system will be provided from that application. This application will have an included text editor the purpose of a special data type that is planned to use in the communications and signature work. Biometrix Web will be designed for the management of all sub-systems and generating transaction report for forensics cases.

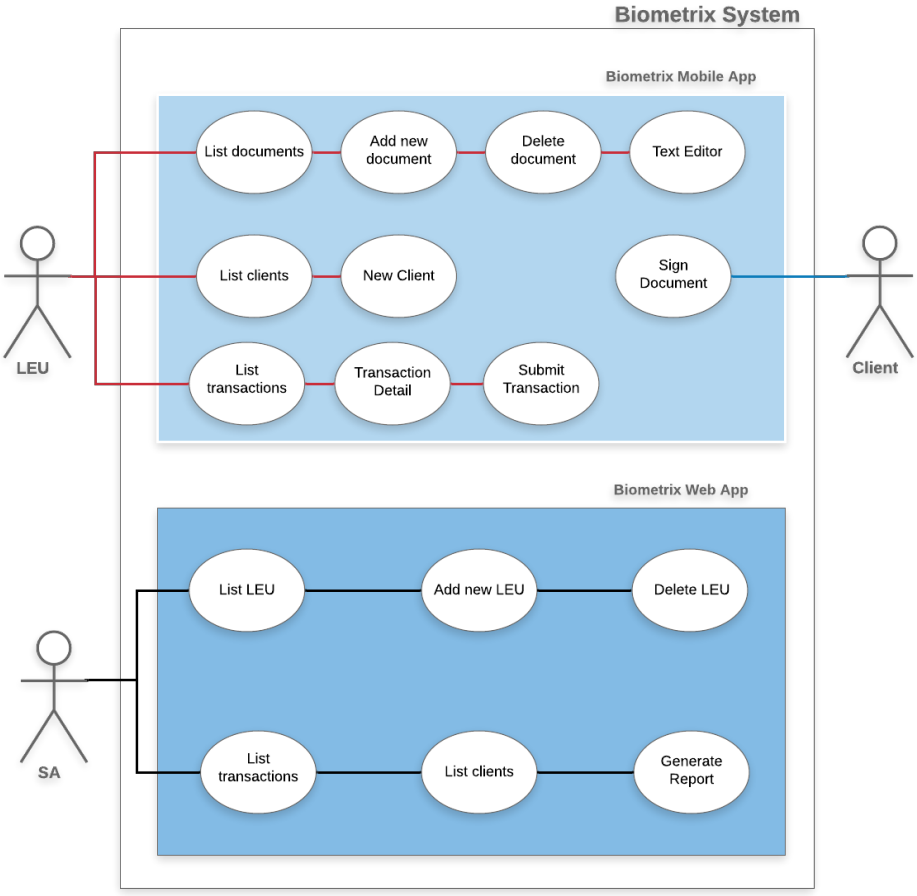


Figure 1: Context Diagram of Biometrix System

5.2.2 Design Elements

Use Case	Description	Actor
Sign a document	To sign any document chosen by LEU expects identification of signer and verification of signature.	Client
List documents	To display workspace of sub-system	LEU
Add new document	To creating a new document with document context using by text-editor	LEU
Delete document	To delete document from workspace choose by LEU	LEU
List Clients	To list all signer accounts in the system	LEU
Register new client	To create a new client, names, signature and id details are must be declared. To enroll new signature, system needs 5 different sample taken by client.	LEU
Monitor transactions	To list all transactions and its detailed information	LEU - SA
List LEUs	To list all LEUs in the sub-system	SA
Add new LEU	To add new LEU account identity information	SA
Delete LEU	To activate/deactivate a LEU in the sub-system	SA
Generate report	To generate a report. This report will include transferred file's hash , signer's details, timestamps etc.	SA

5.2.3 Example Languages

The diagram given in the previous subsections is created by the UML. Use-case diagram is used to establish the context and boundaries of the system.

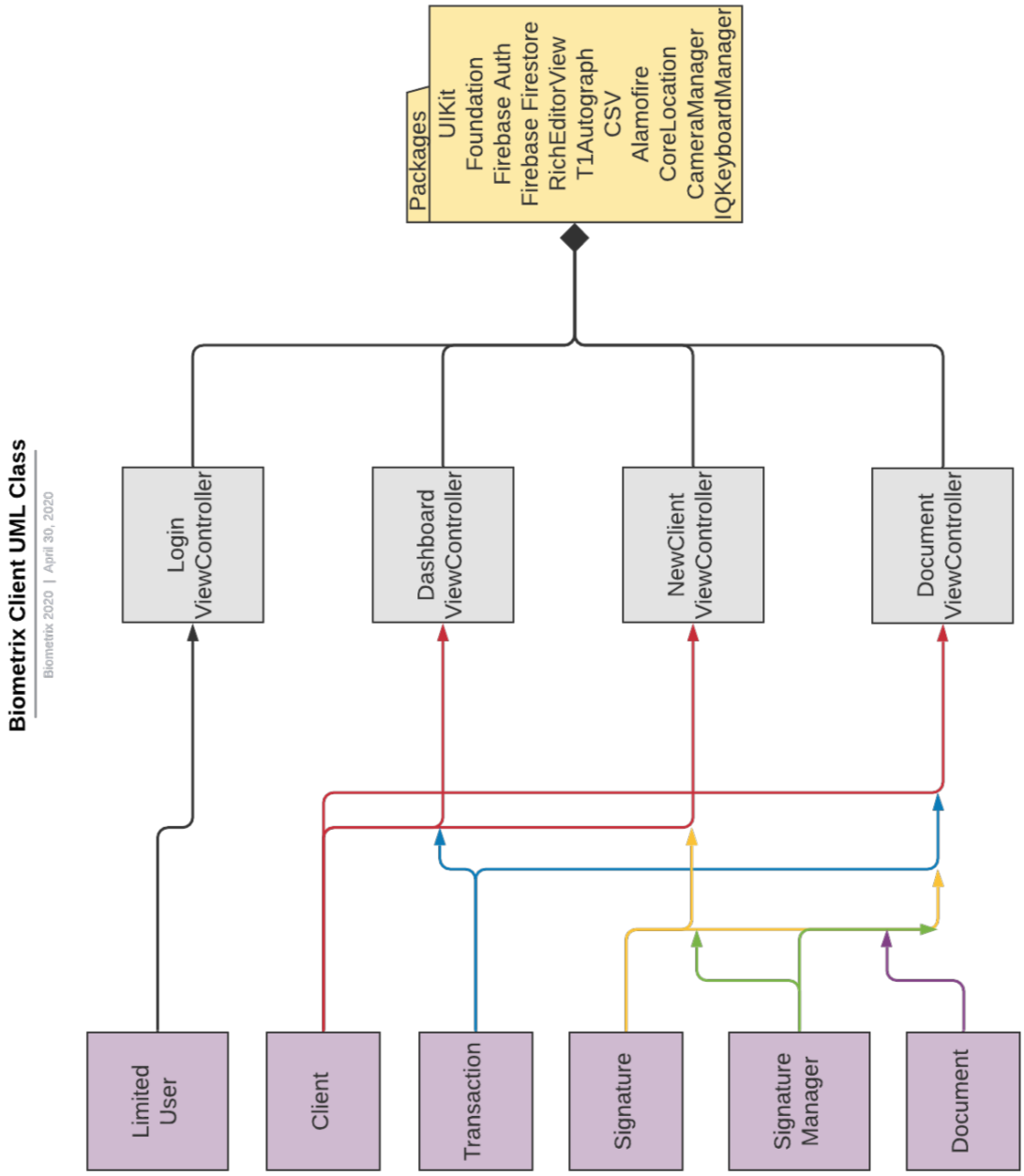


Figure 2. Class Diagram for *biometrix-Client*

5.3 Logical Viewpoint

The purpose of the logical viewpoint is to define classes and interfaces with their structural relationships. For each controller, there will be a diagram to overview the entity and methods. Also, the definition of each element is provided.

5.3.1 Design Elements

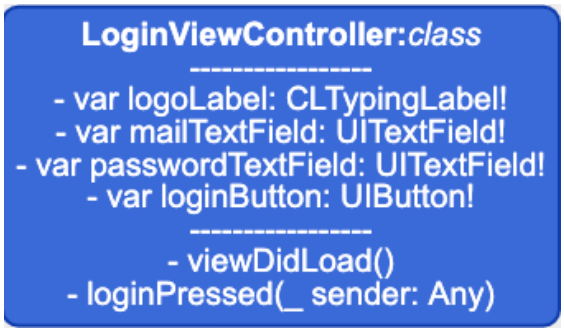
The project has four sub-systems called '*biometrix*-**Client**', '*biometrix*-**Admin**' and '*biometrix*-**Server**'. This revision of the document has only **'Client'** subsystem definitions. **'Admin'** and **'Server'** sub-systems definitions.

5.3.1.1 Biometrix Client

5.3.1.1.1 LoginView Controller

This class defined for authentication of LEU. It imports Firebase Auth SDK.

Diagram



Description:

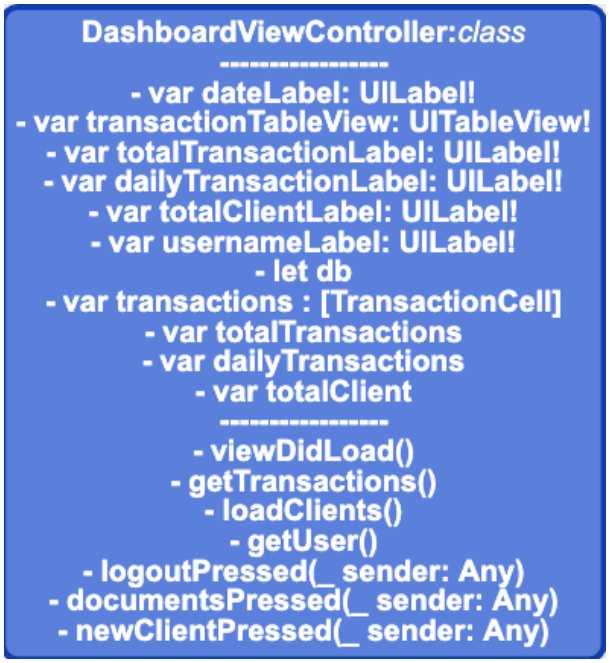
Name	Type/ReturnType	Definition
logoLabel	CLTypingLabel	This variable refers to logo label. It imports CLTypingLabel package for animation

mailTextField	UITextField	This variable refers to get email of the LEU
passwordTextField	UITextField	This variable refers to get password of the LEU
loginButton	UIButton	This variable refers to login button outlet
loginPressed()	Bool	If authentication is succeed, the app segues to *Dashboard* view

5.3.1.1.2 DashboardView Controller

This class defined for control segues. It also provides the latest transaction information as a table view and some information about the system.

Diagram



Description

Name	Type/ ReturnType	Definition
transactionTableView	UITableView	This variable shows the latest transaction as a table view.
dateLabel	UILabel	This variable shows the current date to user
totalTransactionLabel	UILabel	This variable shows the total number of transactions to user
totalClientLabel	UILabel	This variable shows the total number of clients to user
usernameLabel	UILabel	This variable shows the current date to user
totalTransactions	Integer	Counter for total number of transactions
dailyTransactions	Integer	Counter for number of daily transactions
totalClient	Integer	Counter for total number of clients
db	Firestore	This variable refers to Firestore database
transactions	Transaction Cell Array	This variable stores each cell with its transaction data
getTransactions()	-	Gets all transactions from FirestoreDB, reloads documents tableview and updates transaction labels

loadClients()	-	Gets all clients from FirestoreDB and updates total clients label
getUser()	-	Gets full name of authenticated user and updates username label
newclientPressed()	-	This method directs NewClientView
transactionsPressed()	-	This method directs TransactionView
logoutPressed()	-	This method logs out the authenticated user.

5.3.1.1.3 DocumentView Controller

This class defined for document and transaction operations.

Diagram:

Name	Type/ ReturnType	Definition
editorView	RichEditorView	This variable defined for initialize rich text editor.
documentNameTextField	UITextField	This variable gets the name of selected document and saves with new name
clientIDTextField	UITextField	This variable gets client id from user for signing
documentTableView	UITableView	This variable shows the documents from workspace as a table view.

signatureImageView	UIImage View	This variable shows signature of the client before make transaction
resultImageView	UIImage View	This variable shows an image for result of transaction
cameraImageView	UIImage View	This variable defined for hidden camera
signerImage	UIImage	This variable stores image that captured from hidden camera
clientNameLabel	UILabel	This variable shows the full name of client before submit transaction
docNameLabel	UILabel	This variable shows the name of document before submit transaction
transactionDateLabel	UILabel	This variable shows the date of transaction before submit transaction
scoreLabel	UILabel	This variable shows the score of transaction before submit transaction
thresholdLabel	UILabel	This variable shows the threshold of client before submit transaction
differenceLabel	UILabel	This variable shows the difference between score and threshold of client before submit transaction

resultLabel	UILabel	This variable shows the result of transaction before submit transaction
transactionButton	UIButton	This variable defined for disable transaction button before signing
db	Firestore	This variable refers to Firestore database
autographModal	T1 Autograph	This variable defined as T1Autograph struct for signature capturing
signatureManager	Signature Manager	This variable defined for init SignatureManager struct for signature modeling
cameraManager	Camera Manager	This variable defined for init Camera Manager object
locationManager	CLLocation Manager	This variable defined for getting coordinates of device
toolbar	RichEditor Toolbar	This variable defined for Toolbar options for editor-view
html	String	This variable stores content of selected document with style
documents	Document Array	This variable defined for document array for table-view

selectedDoc	Document	This variable stores selected document
transaction	Transaction	This variable defined for transaction request to server as a Transaction struct
latitude	CLLocation Degree	This variable defined for store latitude information of device
longitude	CLLocation Degree	This variable defined for store longitude information of device
updateTransacionUI()	-	This function runs after transaction and it updates data of transaction-card
clearEditor()	-	This function clears editor-view
clearTransactionDetail()	-	This function clears transaction-card
getDocuments()	-	This function gets all documents from FirestoreDB and reloads documents tableview
getClient()	-	This function gets client data from Id and updates UI
createTransaction()	-	This function posts new transaction to DB
createDocument()	-	This function posts new document to DB

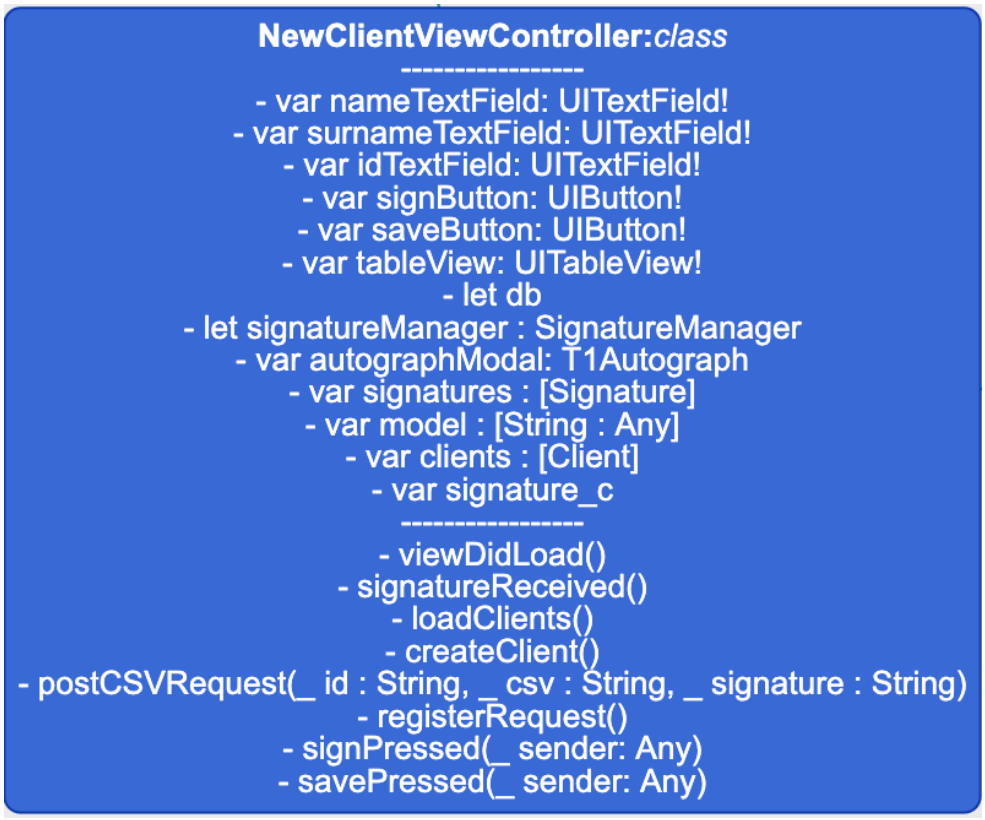
deleteDocument()	-	Delete selected document from DB
getResultRequest()	-	Makes GET request to Biometrix-Data API. /verification?client=
postCSVRequest()	-	Makes POST request to Biometrix-Data API. /exportCSV
postTransactionRequest()	-	Makes POST request to Biometrix-Data API. /transaction
uploadImage()	-	Uploads signature image to Biometrix-Data API. /uploadImg.
uploadSignerImage()	-	Uploads signer image to Biometrix-Data API. /uploadSignerImg.
uploadPressed()	-	This function is for post new document to DB action
newDocumentPressed()	-	This function is for clear editor view
deleteDocumentPressed()	-	This function is for delete document from DB action
signDocumentPressed()	-	This function is for set modal-view for signature capturing
submitTransactionPressed()	-	This function is for confirm transaction action


DocumentViewController:class
- var editorView: RichEditorView!
- var documentNameTextField: UITextField!
- var clientIDTextField: UITextField!
- var tableView: UITableView!
- var signatureImageView: UIImageView!
- var clientNameLabel: UILabel!
- var docNameLabel: UILabel!
- var transactionDateLabel: UILabel!
- var scoreLabel: UILabel!
- var thresholdLabel: UILabel!
- var differenceLabel: UILabel!
- var resultImageView: UIImageView!
- var resultLabel: UILabel!
- var transactionButton: UIButton!
- var cameraImageView: UIImageView!
- let db
- var autographModal: T1Autograph
- let signatureManager : SignatureManager
- let cameraManager
- let locationManager
- var html : String
- var signerImage
- var documents : [Document]
- var selectedDoc
- var transaction : Transaction?
- var latitude : CLLocationDegrees
- var longitude : CLLocationDegrees
- var toolbar: RichEditorToolbar
- let toolbar
- viewDidLoad()
- updateTransactionUI(_ t : Transaction)
- clearEditor()
- clearTransactionDetail()
- getDocuments()
- getClient(_ id : String)
- createTransaction()
- createDocument()
- deleteDocument()
- getResultRequest(_ param : [String : String], _ request : String, _ s : Signature)
- postCSVRequest(_ id : String, _ csv : String, _ signature : String, _ sign : Signature)
- postTransactionRequest()
- uploadImage()
- uploadSignerImage()
- uploadPressed(_ sender: UIButton)
- newDocumentPressed(_ sender: Any)
- deleteDocumentPressed(_ sender: Any)
- signDocumentPressed(_ sender: Any)
- submitTransaction(_ sender: Any)

5.3.1.3 NewView Controller

This class defined for register new client information and signature. Controller expects five signature from client. If registering process success without any errors, it directs DashboardView.

Diagram:



Description:

Name	Type/ReturnType	Definition
nameTextField	TextField	This variable is for get name of new client from user
surnameTextField	TextField	This variable is for get surname of new client

idTextField	TextField	This variable is for get of new client
signButton	UIButton	This variable is for activate/deactivate sign button
saveButton	UIButton	This variable is for activate/deactivate save button
clientTableView	UITableView	This variable is for show the clients from workspace as a table view.
db	Firestore	This variable refers to Firestore database
signatureManager	Signature Manager	This variable defined for init SignatureManager struct for signature modeling
autographModal	T1Autograph	This variable defined as T1Autograph struct for signature capturing
signatures	Signature Array	This variable is defined for store signature array using for client's model
model	String:Any Dictonary	This variable is defined for Stores client's model

clients	Client Array	This variable stores all client information in system
signatureCounter	Integer	This variable counts number of signature that put by client
signatureReceived()	-	This function is for count number of signatures
loadClients()	-	This function is for get all clients from FirestoreDB and reloads clients tableview
createClient()	-	This function is for post new client to DB
CSVRequest()	-	This function is for make POST request to Biometrix-Data API. /exportCSV.
registerRequest()	-	This function is for make GET request to Biometrix-Data API. /register?client=
signPressed()	-	This function is for confirm captured signature
savePressed()	-	This function is for save new client to DB action

5.3.1.2 Biometrix Web-Admin

5.3.1.2.1 Dashboard

Name	Type/ ReturnType	Definition
firebaseConfig	Dictionary	This variable is defined to store configuration informations for database
transactionConverter	Firebase Data Converter	This variable is defined for parsing transaction data
db	Firestore	This variable refers to Firestore database
totalClient	-	This variable is defined for count number of clients
totalTransaction	-	This variable is defined for count number of transactions
totalUser	-	This variable is defined for count number of users
getStats()		This function is defined for get some information about system from DB
getLatestTransactions()		This function is defined for get latest transaction from DB

5.3.1.2.2 Users

Name	Type/ReturnType	Definition
firebaseConfig	Dictionary	This variable is defined to store configuration informations for database
userConverter	Firebase Data Converter	This variable is defined for parsing user data
db	Firestore	This variable refers to Firestore database
newUser()	-	This function creates new user and allow authentication access for specified email and password
updateUI()	-	This function runs when new user registered
deleteUser()	-	This function deletes user

5.3.1.2.3 Transaction

Name	Type/ReturnType	Definition
firebaseConfig	Dictionary	This variable is defined to store configuration informations for database
transactionConverter	Firebase Data Converter	This variable is defined for parsing transaction data
db	Firestore	This variable refers to Firestore database
url	String	This variable refers to url of data api
transactionList	UL element	This variable refers to transaction list as an html element
getTransactions()	-	This function gets all transactions and details and sets UL element
downloadReport()	-	This function makes GET request to data api and downloads transaction analysis

5.3.1.3 Biometrix Data-API

Biometrix Data-API is for manage signature verification and blockchain process for using from Biometrix Mobile Application.

5.3.1.3.1 Routes

Routes file manages requests and libraries. It uses following libraries:

- Flask
- Zipfile
- OS
- Threading

Name	Type	Parameter	Parameter Type
/register	GET	Client	String
/verification	GET	Client	String
/transaction	POST	Transaction	JSON
/exportCSV	POST	Client Signature	String CSV
/uploadSignatureImg	POST	Signature	File
/uploadSignerImg	POST	Signer	File
/downloadReport	GET	Transaction Client	String String

5.3.1.3.2 Extract

Extract file is used for extracting local features for each point from CSV formatted signature file. After extraction saves as CSV formatted in clients model directory. It uses following libraries:

- Numpy
- Pandas

Name	Parameter	Definition
init()	Client ID	This function runs all necessary functions in order
normalize_location()	DataFrame	This function defined for normalize x - y coordinates
smooth()	DataFrame	This function defined for smooth all x - y coordinates of signature
extract()	DataFrame	This function runs all extraction functions in order

magnitude()	DataFrame	This function calculates magnitude value of two vector
derivative()	DataFrame	This function calculates first derivative of the vector
alfa()	DataFrame	This function calculates Angle(α) between absolute velocity vector and x axis
sinAngle()	DataFrame	This function calculates Sin(α)
cosAngle()	DataFrame	This function calculates Cos(α)
beta()	DataFrame	This function calculates Angle(β) between two adjacent line segments at each coordinate
feature_normalization()	DataFrame	This function defined for Z-score normalization

5.3.1.3.3 Model

Model file generates signature model and calculates threshold. It uses following libraries:

- Pandas
- DTW
- OS

Name	Parameter	Definition
generate()	Client ID Threshold	This function generates client model which contains id, threshold and signature raw data
calc_threshold()	Client ID	This function calculates threshold value and exports as CSV file

5.3.1.3.4 Blockchain

Blockchain file generates blockchain files as JSON formatted. It uses following libraries:

- Hashlib
- Json
- OS
- Time

Name	Parameter	Definition
write_block()	Model	This function generates blockchain model for new block using other methods

get_POW	Index	This function calculates proof of work of specified block
is_valid_proof	Last_proof Proof Difficulty	This function helps to get_POW function for calculating proof of work
get_next_block	-	This function gets next index of blockchain
get_hash	File name	This function is calculates hash of specified file using SHA-256

5.3.1.3.5 Transaction

Transaction file generates transaction files and their analysis. It uses following libraries:

- Json
- OS
- Pandas
- DTW
- Shutil

Name	Parameter	Definition
generate()	Transaction Data	This function generates detail.json file which contains transaction details

analysis()	Client ID	This function plots analysis between signature and model for each local feature
csv_to_json()	File Path	This function converts CSV file to JSON and exports

5.3.1.3.5 Verification

Verification file calculates similarity between model and signature and makes a decision using calculated score . It uses following libraries:

- Numpy
- Pandas
- DTW

Name	Parameter	Definition
get_result()	Client ID	This function calculates similarity between model and signature and returns a decision result using calculated score

5.3.2 Example Languages

A class diagram that describes the structure of a system by showing classes of the system has given in the previous sections by using UML modeling language.

5.4 Dependency Viewpoint

The dependency viewpoint of Biometrix system includes shared information, execution processes, and system calls. It explains access routes and how data is transferred from scratch between Biometrix subsystems.

5.4.1 Design Concerns

Biometrix ecosystem consists of 3 main subsystems: End-User, Server, and Database. There is no connection between server and database. The server has file storage independent from the database. This design server just focusses on machine learning and storing sensitive data. Database subsystem stores other information related to clients, transactions, documents, and users.

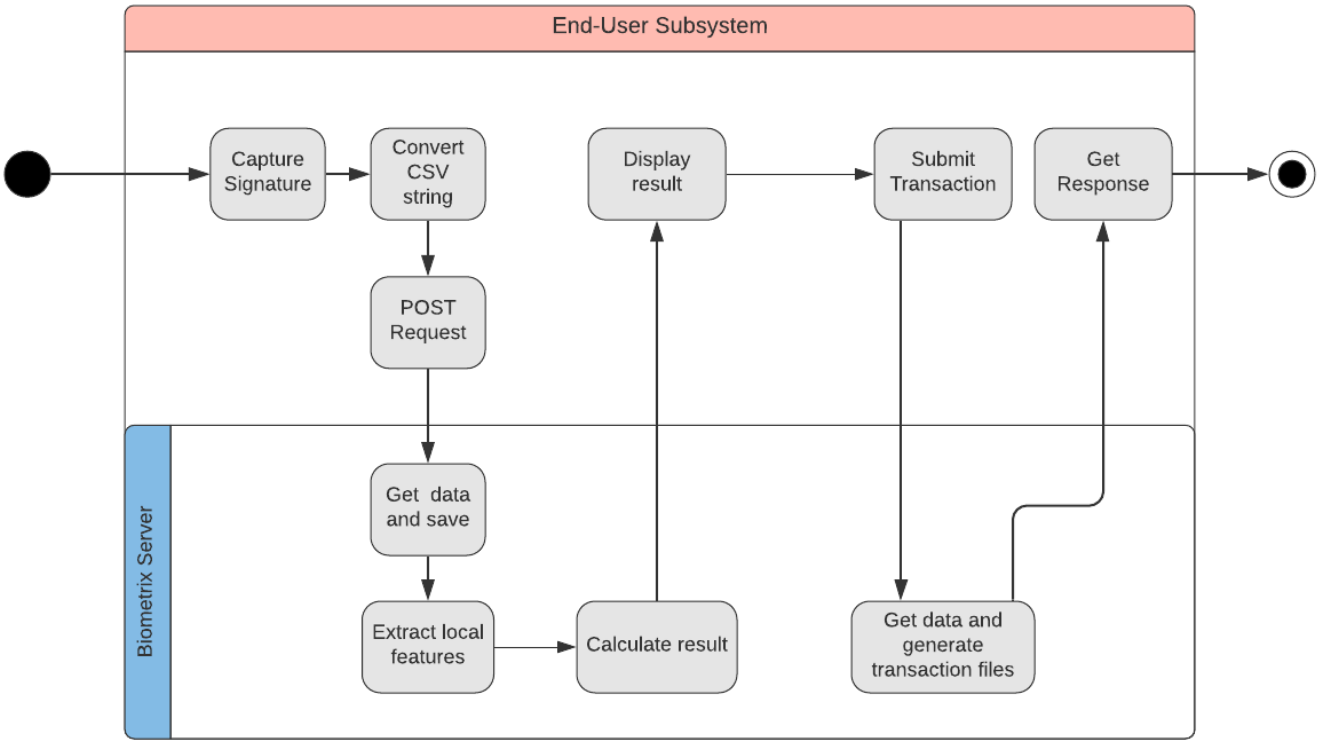


Figure 3: Activity Diagram for a transaction

This figure shows how transactions occur on the server-side. Biometrix collects signature data of clients using end-user assigned mobile tablet devices. After taking signature, the mobile app generates a model, converts to CSV string, and makes a post request to Biometrix-Server. The data contains client information, signature data, timestamp, etc. Processing is managed by a Python Rest API application using the Flask micro-framework. After receiving data, the application parses the data and calls the necessary functions defined previously. When calculations are done, returns a result of the transaction before confirmed by LEU.

5.5 Information Viewpoint

Information viewpoint of Biometrix system includes data types and purposes to use those types in Biometrix ecosystem.

5.5.1 Design Concerns

Biometrix system manages data in several data types: JSON, CSV, NoSQL, JPEG Those data are kept in two separate storages: for personal data: NOSQL Firestore database and for sensitive data: JSON and CSV. Data transfers from the end-user subsystem to server subsystem managed using JSON format for simplicity and prevalence. So, any data to be transferred has to be converted to JSON.

Following tree-like file directory demonstrates an example transaction structure:

```
├── 5DC8223D-CC87-4371-877B-8F1269F987F6
│   ├── details.json
│   ├── signature.csv
│   ├── signer.jpeg
│   ├── analysis
│           ├── 1
│           ├── 2
│           └── 3
│           └── 4
│           └── 5
```

5.5.2 Design Elements

Data Type	Purpose	Description
JSON	Data for transfer between subsystems	Biometrix system requires arrays of objects to be transferred mostly. Therefore, data needs to be encoded to JSON.
NOSQL	Data for store Firestore database	Biometrix system database is an NOSQL database
JPEG	Data for store signer and signature image files	Biometrix has to keep various types of inseparable data in blockchain. Therefore, a structure is created to store all data required in a single chain.
CSV	Data for signature registration and verification	Biometrix exports signatures as CSV formatted. Feature extraction, verification processes uses this files

5.5.3 Data Attribute

Biometrix uses a JSON format to store personal data in the blockchain. This data structure consists of an identification number of the person in which the data belongs, signature threshold, and signature arrays. Those values are filled with data provided by the user-end subsystem and could not be left empty. After the structure is created and filled, it is pushed to the blockchain.


```
▼ {
  ▼ "data": {
      "client": "1",
      "threshold": 2.0694338110917636,
    ▼ "signatures": {
        ▶ "s1": { 249 items },
        ▶ "s2": { 230 items },
        ▶ "s3": { 207 items },
        ▶ "s4": { 230 items },
        ▶ "s5": { 286 items }
      }
    },
    "prev_hash": "83d7411b8069a6f1ffa6aeee6379f967a2d666eb65cc5cf4eeef318b058c5b55",
    "timestamp": 1586800014.914481,
    "proof": 95,
    "index": "2"
}
```


An example blockchain file. (JSON)

Data, such as transactions, mail addresses, names, and surnames, are not critical data and mentioned as impersonal data and these data are stored in the NoSQL database. Signatures, identity numbers, and threshold data are personal data of the user and they are very critical. So, these data have to be stored in a secure environment.

5.5.4 Example Languages

The diagram given in the previous subsections is created by the UML. Activity diagram is used to demonstrate most important data-flow in the system.

5.6 Pattern Viewpoint

It describes design patterns and usage of design patterns in the system.

5.6.1 Design Concerns

In Biometrix Mobile uses MVC pattern which is Apple's preferred way of architecting apps for its platforms.

5.6.2 Design Elements

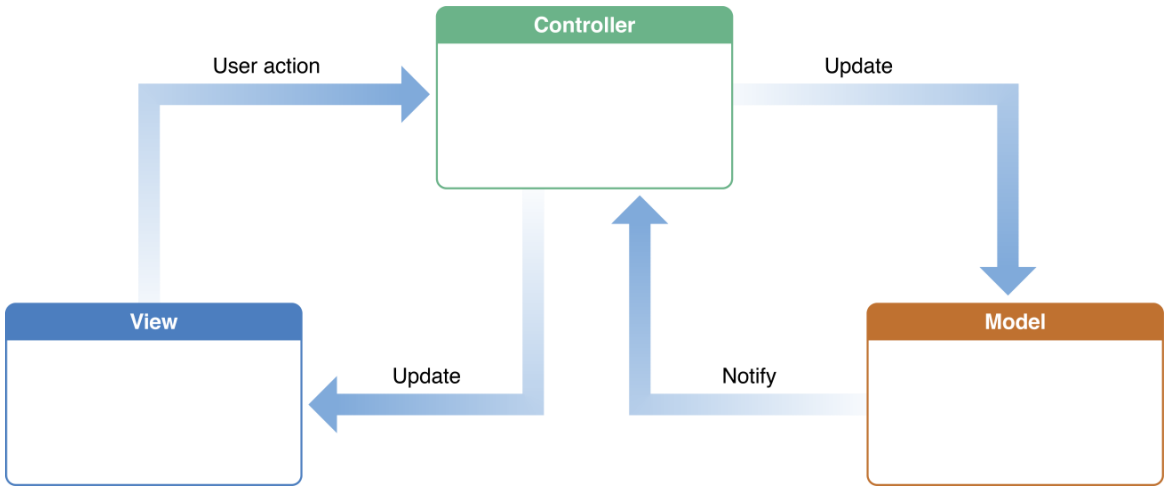


Figure 4. MVC Pattern

The Model layer for the Biometrix login and signature data. The Model layer is responsible for the business logic of the Biometrix application. All data of Biometrix such as signatures and files encapsulated here.

The Model layer is responsible for the business logic of an application. It will encapsulate methods to access data (databases, files, etc.) and will make a reusable class library available.

The View layer is an object that the user of Biometrix can see in a UIThe Controller controls all logic that goes between the View and Model layer.

5.7 Interface Viewpoint

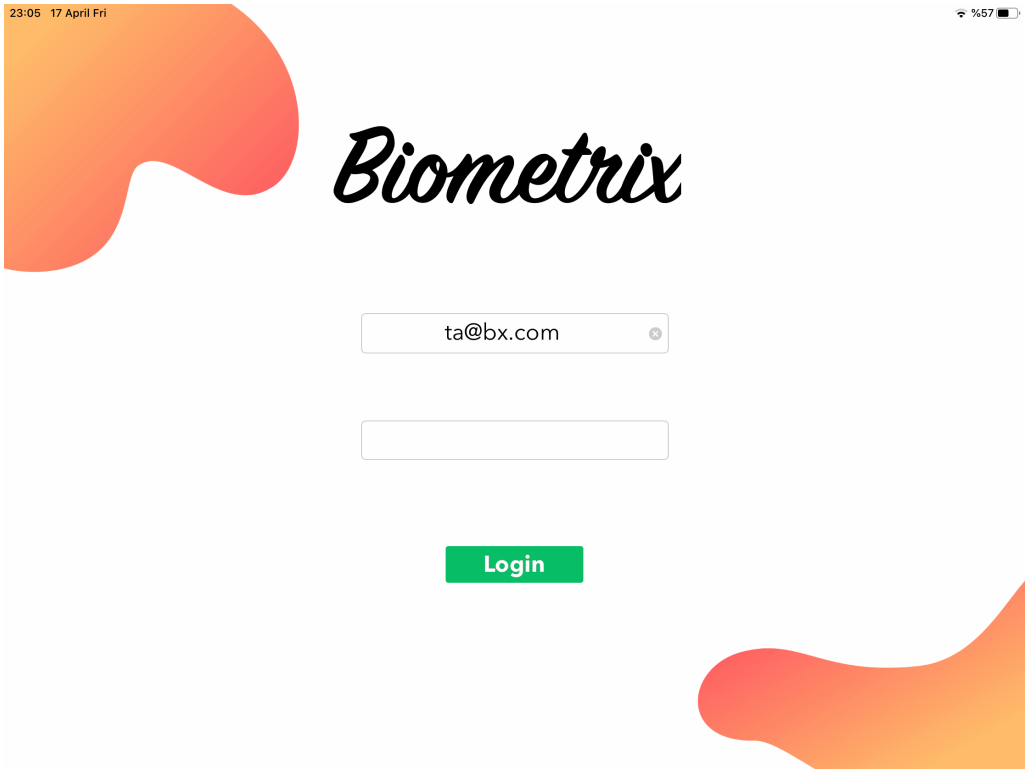
In this part of the document, the details of external and internal interfaces will be defined. There shall be three interfaces in Biometrix, which are Client Mobile App Interface, Admin Mobile App Interface, and Web App Interface. This revision of the document includes only Client Mobile App Interface

5.7.1 Design Concerns

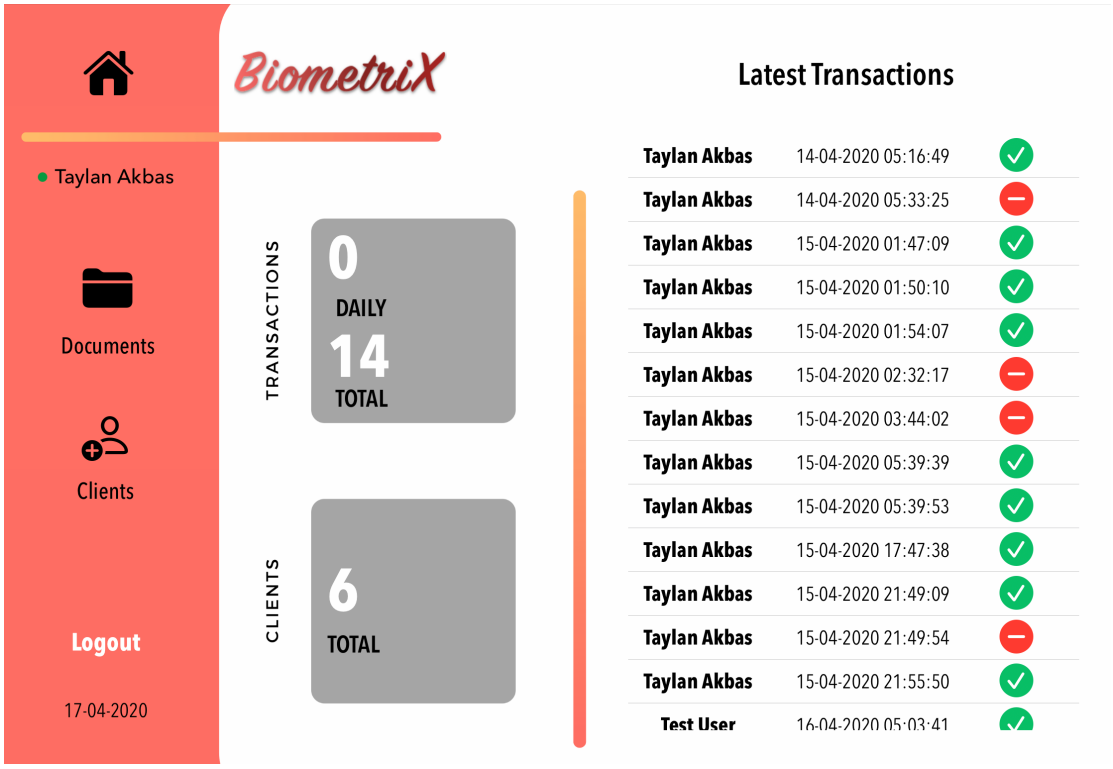
The interface viewpoint provides information to the designers, developers, and testers before proceeding with the detailed design of the system. It also informs them about how cooperating entities will interact. With the ease of each interface description, designers and developers can know the internal and external connections of the system to develop it. This contributes to ease of maintenance.

5.7.2 Design Elements

5.7.2.1 Biometrix Mobile



Login View



Dashboard View

New Client View

Select Document

Document 1

Document 2

Transaction Detail

Client	Document
Test User	Document 1

Timestamp: 03-05-2020 19:15:20

Score: 2.2499745334929253

Threshold: 2.099424979028366

Difference: 0.151

Geniue

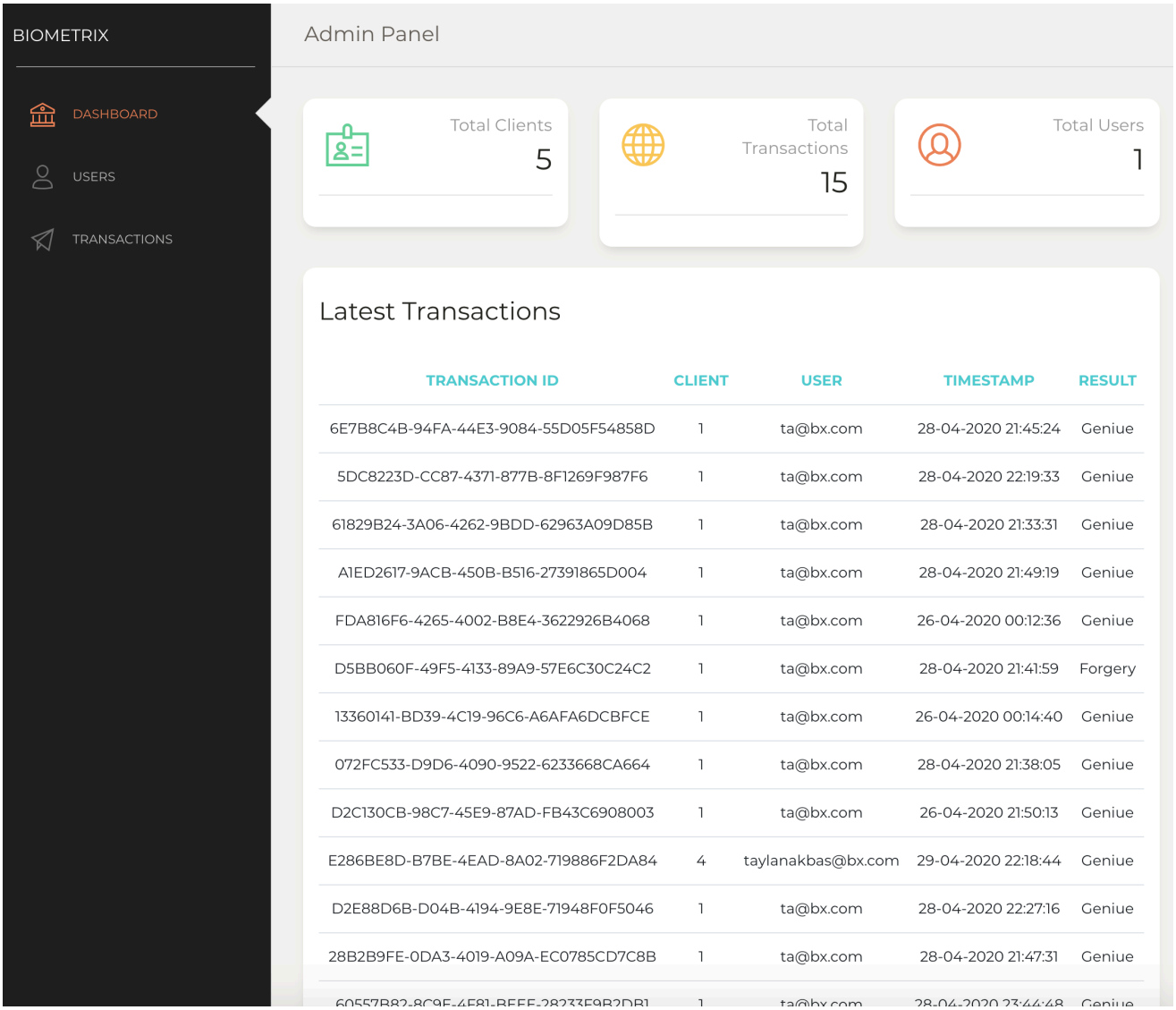
Document	Client
Document 1	10

TITLE

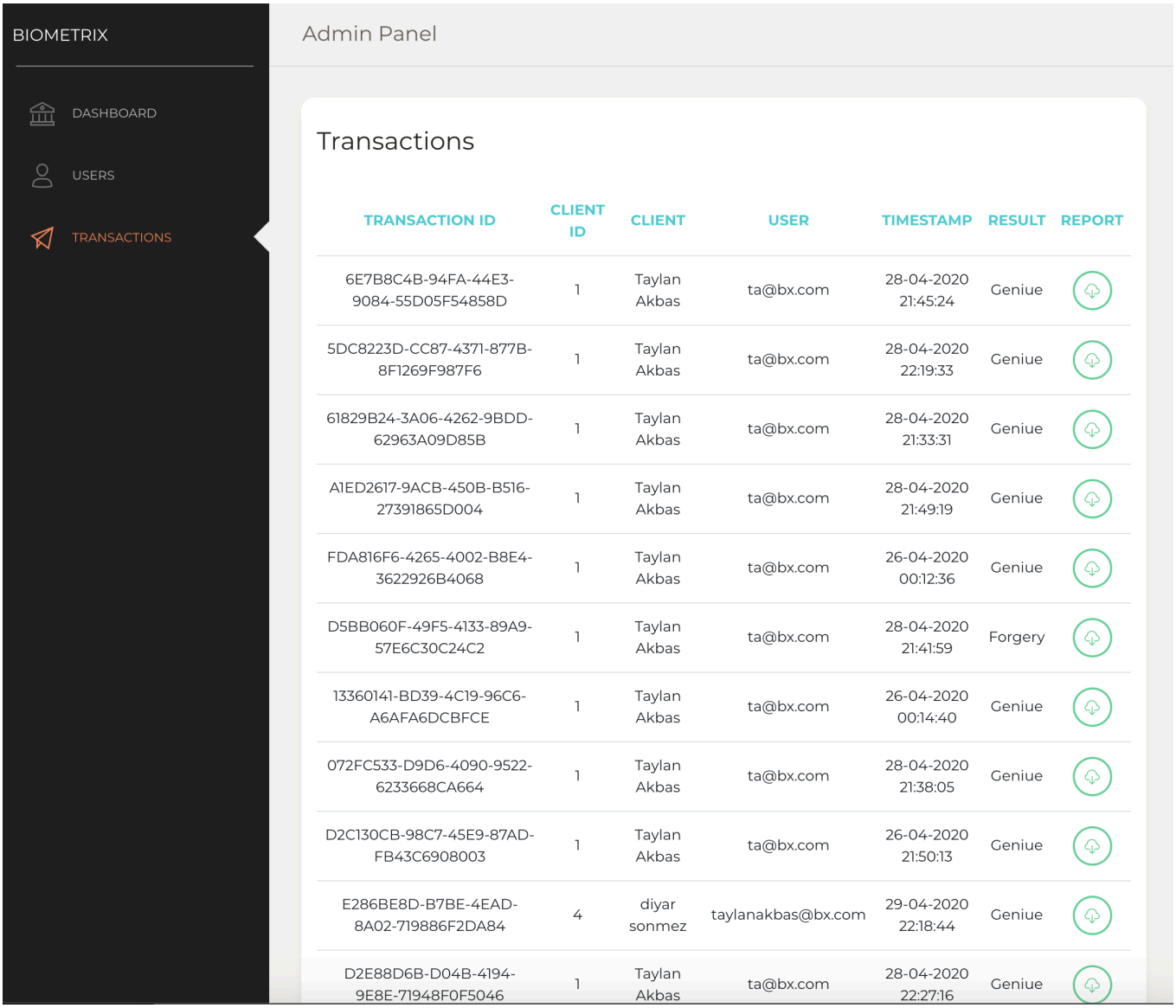
Sed vulputate, lectus et sagittis sodales, ligula massa molestie purus, sit amet vestibulum nulla sem convallis dui. Sed ac bibendum mauris. Phasellus in viverra erat, euismod pellentesque neque. Nullam efficitur eros bibendum cursus scelerisque. Curabitur elementum mi a velit varius porta. Vestibulum dignissim nisi eu odio pellentesque varius. Sed in sapien semper, congue purus et, viverra lacus. Vestibulum eget eros facilisis, bibendum felis non, fermentum nisl. Nullam sed placerat erat. Ut a libero ac nisi feugiat sollicitudin sit amet sit amet mi. Duis vel neque ac risus consequat elementum non ac leo. Nulla pretium tortor a justo hendrerit condimentum. Donec sapien ipsum, interdum non fringilla et, pellentesque non neque. Nunc aliquet neque a sem consequat, id eleifend nulla malesuada. Suspendisse vestibulum posuere dui, et vulputate ipsum porttitor eu. Duis nec augue feugiat arcu interdum faucibus.

Documents View

5.7.2.1 Biometrix Web

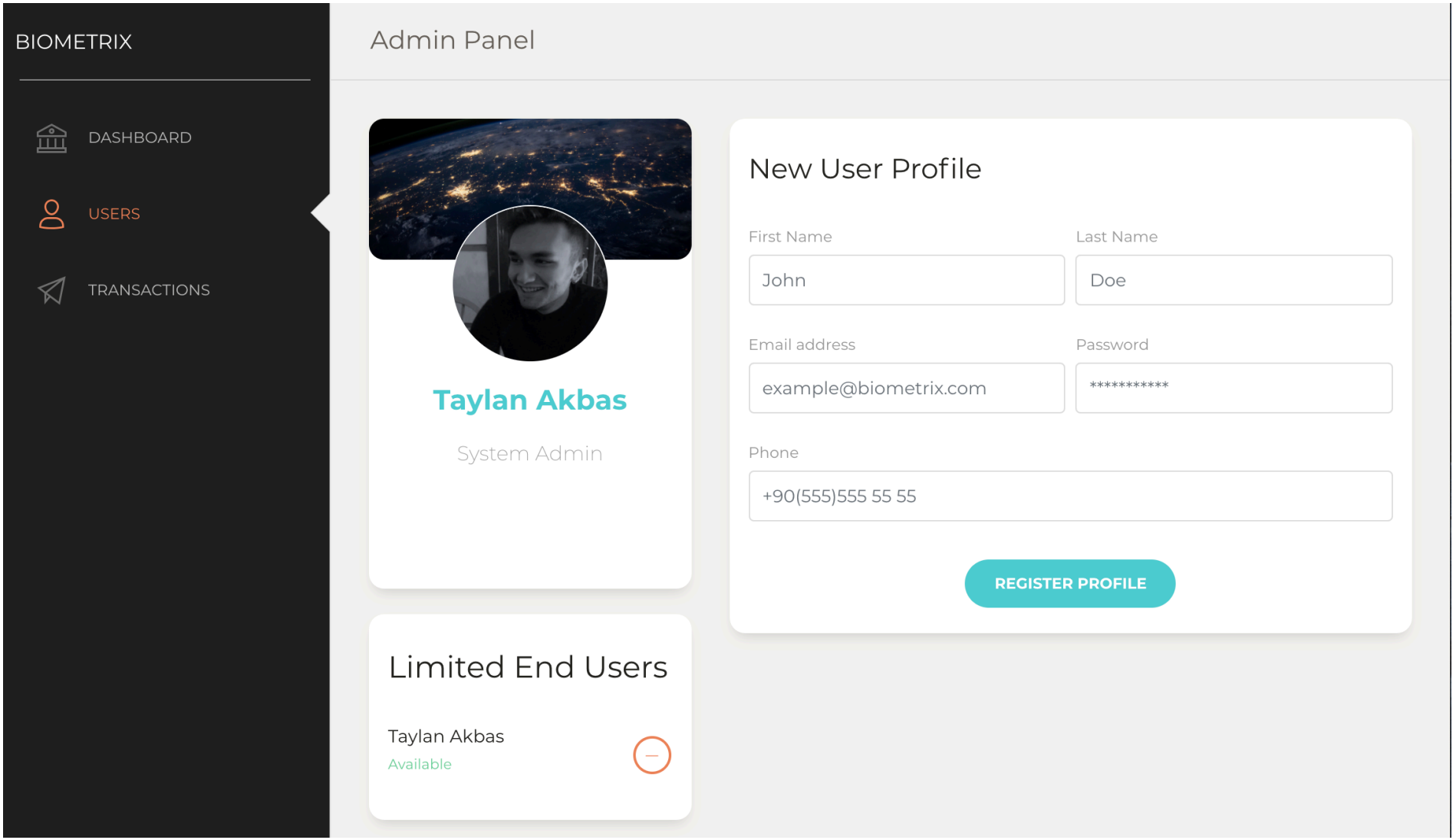


Dashboard View



TRANSACTION ID	CLIENT ID	CLIENT	USER	TIMESTAMP	RESULT	REPORT
6E7B8C4B-94FA-44E3-9084-55D05F54858D	1	Taylan Akbas	ta@bx.com	28-04-2020 21:45:24	Geniue	
5DC8223D-CC87-4371-877B-8F1269F987F6	1	Taylan Akbas	ta@bx.com	28-04-2020 22:19:33	Geniue	
61829B24-3A06-4262-9BDD-62963A09D85B	1	Taylan Akbas	ta@bx.com	28-04-2020 21:33:31	Geniue	
A1ED2617-9ACB-450B-B516-27391865D004	1	Taylan Akbas	ta@bx.com	28-04-2020 21:49:19	Geniue	
FDA816F6-4265-4002-B8E4-3622926B4068	1	Taylan Akbas	ta@bx.com	26-04-2020 00:12:36	Geniue	
D5BB060F-49F5-4133-89A9-57E6C30C24C2	1	Taylan Akbas	ta@bx.com	28-04-2020 21:41:59	Forgery	
13360141-BD39-4C19-96C6-A6AFA6DCBFCE	1	Taylan Akbas	ta@bx.com	26-04-2020 00:14:40	Geniue	
072FC533-D9D6-4090-9522-6233668CA664	1	Taylan Akbas	ta@bx.com	28-04-2020 21:38:05	Geniue	
D2C130CB-98C7-45E9-87AD-FB43C6908003	1	Taylan Akbas	ta@bx.com	26-04-2020 21:50:13	Geniue	
E286BE8D-B7BE-4EAD-8A02-719886F2DA84	4	diyar sonmez	taylanakbas@bx.com	29-04-2020 22:18:44	Geniue	
D2E88D6B-D04B-4194-9E8E-71948F0F5046	1	Taylan Akbas	ta@bx.com	28-04-2020 22:27:16	Geniue	

Transaction View


BIOMETRIX
DASHBOARD
USERS
TRANSACTIONS
Admin Panel
Taylan Akbas
System Admin
Limited End Users
Taylan Akbas
Available
New User Profile
First Name
John
Last Name
Doe
Email address
example@biometrix.com
Password
****
Phone
+90(555)555 55 55
REGISTER PROFILE

Users View

APPENDIX C: PRODUCT MANUAL

Yasar University

COMP4920

Senior Design Project II, Spring 2020

Advisor: *Ahmet Koltuksuz, Ph.D., Assoc. Prof.*

Product Manual

for

BiometriX

Artificial Intelligence Assisted Biometric Signature on Block Chain

Revision 2.0

Prepared by:

Taylan Akbaş

1507001032

This page intentionally left blank.

TABLE OF CONTENTS

Revision History

Revision	Date	Explanation
1.0	13.04.2020	Defined Biometrix Mobile App implementation and source code organization
2.0	04.05.2020	Finalized the document

1. Introduction

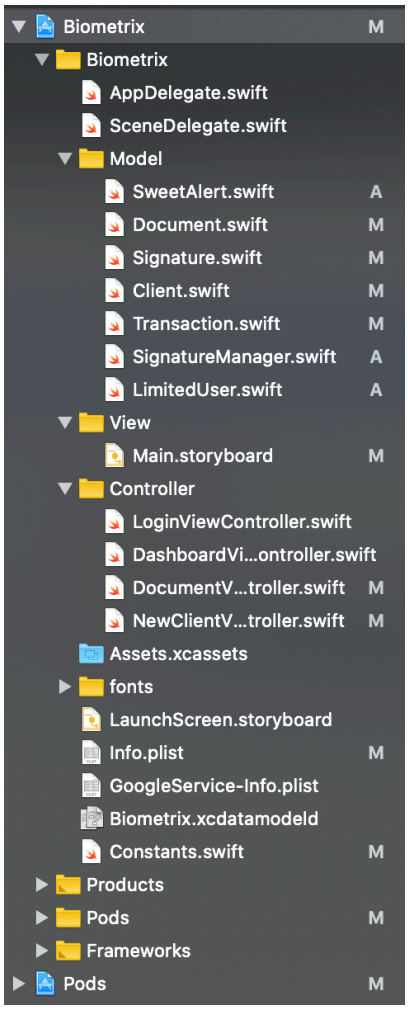
The purpose of this document is to explain implementation, development tools, hardware and software platform and installation for each subsystem. Biometrix system implemented as it is described in Biometrix Design Specification Document Revision, 2.0 and Biometrix Requirements Specification Document, Revision 1.0

2. Biometrix Software System Implementation

2.1 Biometrix Mobile App Implementation

Biometrix is a mobile application for signing any document in an accustomed way with a state of the art level of security. Principally the application focuses on dynamic (online) signature recognition which will be assisted by artificial intelligence for obviating forgeries.

2.1.1 Source Code and Executable Organization



Biometrix Mobile App uses MVC architecture. Models folder contains data models in dictionary type. The data transferred from Firestore DB or Biometrix Data API.

```
├── Model
│   ├── Client.swift
│   ├── Document.swift
│   ├── LimitedUser.swift
│   ├── Signature.swift
│   ├── SignatureManager.swift
│   ├── SweetAlert.swift
│   └── Transaction.swift
```

Controller folder contains ViewController files for each view which are operates necessary services and related front-end operations.

```
├── Controller
│   ├── DashboardViewController.swift
│   ├── DocumentViewController.swift
│   ├── LoginViewController.swift
│   └── NewClientViewController.swift
```

View folder contains XML formatted Main.storyboard file that designed for user interactions with mobile app.

```
├── View
│   └── Base.lproj
│       └── Main.storyboard
```

Assets folder contains all other design elements (images, icons, colors, fonts etc.)

Biometrix has Pods folder. This folder auto-generated by *CocoaPods* according to Podfile. Cocoapods is a dependency manager for Swift projects. All dependencies are under the Pods folder.

```
— Biometrix
├── FirebaseAuth
├── Assets.xcassets
│   ├── AppIcon.appiconset
│   ├── appName.imageset
│   └── logo.imageset
├── Constants.swift
├── Controller
│   ├── DashboardViewController.swift
│   ├── DocumentViewController.swift
│   ├── LoginViewController.swift
│   └── NewClientViewController.swift
├── GoogleService-Info.plist
├── Info.plist
├── Model
│   ├── Client.swift
│   ├── Document.swift
│   ├── LimitedUser.swift
│   ├── Signature.swift
│   ├── SignatureManager.swift
│   ├── SweetAlert.swift
│   └── Transaction.swift
├── SceneDelegate.swift
├── View
│   └── Base.lproj
│       └── Main.storyboard
```

```
├── Pods
    ├── Podfile
    ├── Pods
        └── FirebaseFirestore
        └── FirebaseAuth
        └── IQKeyboardManagerSwift
        └── RichEditorView
        └── T1Autograph
        └── CSV
        └── Alamofire
```

Tree-like formatted source code organization

2.1.2 Software Development Tools

Documentation :

- **Jazzy - Version 0.13.2 - https://github.com/realm/jazzy**

Jazzy is a command-line utility that generates documentation for Swift or Objective C.

IDE :

- **Xcode - Version 11.4 - https://developer.apple.com/xcode/**

Xcode is an integrated development environment (IDE) for macOS containing a suite of software development tools developed by Apple for developing software for macOS, iOS, iPadOS, watchOS, and tvOS.

Backend Services:

- **Firebase - Version 6.22.0 - https://cocoapods.org/pods/Firebase**

Firebase is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. It is built on Google's infrastructure. Firebase is categorized as a NoSQL database program, which stores data in JSON-like documents.

- **Alamofire - Version 5.1.0 - https://cocoapods.org/pods/Alamofire**

Alamofire is a Swift-based HTTP networking library for iOS and macOS. It provides an elegant interface on top of Apple's Foundation networking stack that simplifies a number of common networking tasks.

Dependency Manager:

- **Cocoapods - Version 1.8.4 - https://cocoapods.org/**

CocoaPods is a dependency manager for Swift and Objective-C Cocoa projects.

Dependencies :

- **T1Autograph - Version 2.0.4 - https://cocoapods.org/pods/T1Autograph**

T1Autograph is simple to use, standards-compliant, and available for iOS and Android, and aims to capturing signature. Outputs PNG, PDF, SVG, and ISO/IEC 19794-7 biometric interchange XML formats.

- **RichEditorView - Version 5.0 - https://cocoapods.org/pods/RichEditorView**

RichEditorView is a simple, modular, drop-in UIView subclass for Rich Text Editing.

- IQKeyboardManageSwift - Version 6.5.5 - **https://cocoapods.org/pods/IQKeyboardManager**

While developing iOS apps, we often run into issues where the iPhone keyboard slides up and covers the UITextField/ UITextView. IQKeyboardManager allows you to prevent this issue of keyboard sliding up and covering UITextField/UITextView without needing you to write any code or make any additional setup.

- **CSV.swift - Version 2.4.3 - https://cocoapods.org/pods/CSV.swift**

CSV reading and writing library written in Swift.

VCS :

- Git - Version 2.24.1
- Github Desktop - Version 1.6.5

Inspiration & Design Tools :

- Dribble
- Procreate

2.1.3 Hardware and System Software Platform

Apple iPad Air 3rd Generation

- Versions: A2153 - A2123 - A2152
- Release Year: 2019
- Dimensions: 250.6 x 174.1 x 6.1 mm (9.87 x 6.85 x 0.24 in)
- Resolution: 1668 x 2224 pixels – 10.5 inch
- Memory: 64GB – 3 GB RAM
- Comms: Wi-Fi + Cellular & GPS
- Operating system: iOS 13.4

Apple Pencil Generation I

- Dimensions:
  Length : 6.92 inches (175.7 mm) measured from tip to cap
  Diameter : 0.35 inch (8.9 mm)
  Weight : 0.73 ounce (20.7 grams)
- Connections:Lightning connector – Bluetooth

Apple iMac (Retina 4K, 21.5-inch, 2017)

- Processor: 3.4 Ghz Quad-Core Intel Core i5
- Memory: 8 GB 2400 MHz DDR4
- Graphics: Radeon Pro 560 4GB
- Operating System: macOS Catalina 10.15.4

2.1.4 Installation

To install Biometrix iOS13

1. Open Terminal
2. Type following command:
3. git https://github.com/taylanakbas/Biometrix-iOS13.git
4. Open *Biometrix.xcworkspace* from project directory with Xcode
5. Plug your iPad into your computer and select your device top of the Xcode
6. Unlock your device and (⌘R) run the application

2.1.4 Usage

You can log-in to application with following credentials:

E-mail : taylanakbas@bx.com

Password : 123456

2.2. Biometrix Data API Implementation

Biometrix Data-API is for manage signature verification and blockchain process for using from Biometrix Mobile Application. This API has several features:

- Register new client with signature and personal information.
- Verify client signature using model.
- Generate transaction file which consists detailed information about verification.
- Simple blockchain structure as a proof of concept.
- Export signature.csv file.
- Upload signer & signature image collected.
- Generate transaction report including analysis figures.

All data that needs Data-API is collected from Biometrix iOS13 Application.

2.2.1 Source Code Organization

```
├── app
│   ├── __init__.py
│   ├── blockchain.py
│   ├── extract.py
│   ├── model.py
│   ├── routes.py
│   ├── transaction.py
│   ├── util.py
│   └── verify.py
├── biometrix.py
├── blocks
│   └── 1.json
└── clients
    └── 1
        ├── s0.csv
        ├── s2.csv
        ├── s3.csv
        ├── s4.csv
        ├── s5.csv
        └── transactions
```

Tree-like formatted source code organization

2.2.2 Software Development Tools

IDE :

- **Jupyter Notebook - Version 6.0.3 - https://github.com/jupyter/notebook**:

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.

REST Client :

- **Postman - Version 7.23.0 - https://www.postman.com/**

Postman is a collaboration platform for API development. Postman's features simplify each step of building an API and stream

2.2.3 Installation

This instructions explains all requirements and how to test the Biometrix Data-API.

First, you need to install Anaconda Virtual Environment for **Python 3.7**.

You can find installation instructions for your operating system

- Install Conda After install conda, check your version:

```
conda --version
```

Now, you are ready for create virtual environment for Biometrix Data-API.

```
conda create -n bxvenv python=3.7
```

To activate your environment:

```
conda activate bxvenv
```

Installing Packages

```
pip install flask
pip install numpy
pip install pandas
pip instal matplotlib
pip install dtw-python
```

2.2.4 Usage

First you need to get Biometrix Data-API:

```
git https://github.com/taylanakbas/Biometrix-Data-API.git
```

Go to project directory from terminal and type following command:

```
flask run
```

Now the API is running on localhost:5000.

How to use the API

You can use any tool for calling http methods to get the response. I will recommend you to use Postman.

Structure of URI:

- **http://localhost:5000/exportCSV | POST**

This URI is used via POST method to send a JSON containing all information needed for adding new signature data for a client. Example data and format is:


```
{
   "client": "1",
   "csv":
"id,x,y,p,ax,ay,vx,vy,altidude,azimuth,timestamp\n100,140.6376953125,317.1900939941406,0.08,0.0,0.0,0.0,0.0,0.7643388,1.6345018,535535.710999999999\n101,140.97705078125,317.1900939941406,0.08,0.298546315807993,0.0,0.0035825558208367235,0.0,0.7643388,1.6345018,535535.723\n102,142.164794921875,317.1900939941406,0.115625,0.3099096779724234,0.0,0.008851020344058914,0.0,0.7643388,1.6345018,535535.74\n103,146.27099609375,316.9355773925781,0.16665548,1.4787921378467033,-0.12594923003616787,0.03251169446834945,-0.0020151876736580238,0.7643388,1.6345018,535535.75599999
   "sign": "1"
}
```

✳ **http://localhost:5000/register | GET**

This URI is used via GET method for

- Extract local features for each signature.csv (You need 5 signature .csv file)
- Calculate threshold value and save threshold.csv
- Generate a model and write a block in blockchain

'clients/1' folder will have five signature file for test.
You can test with following URI: http://localhost:5000/register?client=1

✳ **http://localhost:5000/verification | GET**

This URI is used via GET method for

- Extract local features for test signature
- Get verification result

'clients/1' folder already includes signature0.csv file for test.
You can test with following URI: http://localhost:5000/verification?client=1

Verification result data and format will be:


```
{
   "dif": "-0.031",
   "result": "Geniue",
   "score": "2.0679527815676306",
   "threshold": "2.099424979028366"
}
```

✳ **http://localhost:5000/transaction | POST**

This URI is used via POST method to send a JSON containing all information needed for adding new transaction data for a client. Example data and format is:


```
{
   "dif": "-0.031",
   "client": "1",
   "score": "2.0679527815676306",
   "user": "user@bx.com",
   "result": "Geniue",
   "longitude": 27.18532879991539,
   "document": "08517ABE-263C-4694-AC9A-3598C58EF8E0",
   "timestamp": "03-05-2020 00:51:58",
   "latitude": 38.455414415892214,
   "transaction": "2070A241-EDB2-45F4-A2CD-BCD2A6949E52",
   "threshold": "2.099424979028366"
}
```


After make this request, you can examine details.json file in

'clients/1/transactions/2070A241-EDB2-45F4-A2CD-BCD2A6949E52'

✳ **http://localhost:5000/uploadSigner | POST**

This URI is used via POST method to upload signer image in jpeg format.

✳ **http://localhost:5000/uploadSigner | POST**

This URI is used via POST method to upload signature image in jpeg format.

File name must be following format:

- TRANSACTION_ID-CLIENT_ID.jpeg

✳ **http://localhost:5000/downloadReport | GET**

This URI is used via GET method to generate analysis and download.

You can test with following URI: http://localhost:5000/downloadReport?client=1&trid=2070A241-EDB2-45F4-A2CD-BCD2A6949E52

2.3 Biometrix Web App Implementation

Biometrix Web App is implemented for monitoring transactions and user operations by System Administrators. This app has several features:

- Monitor transactions.
- Add new user to database
- Delete user from database
- Generate transaction report

This version of app is not includes many backend concerns like login operation, security issues and error handlings.

2.3.1 Source Code and Executable Organization

```
├── assets
│   ├── css
│   │   ├── bootstrap.min.css
│   │   ├── bootstrap.min.css.map
│   │   ├── paper-dashboard.css
│   │   ├── paper-dashboard.css.map
│   │   └── paper-dashboard.min.css
│   ├── fonts
│   │   ├── nucleo-icons.eot
│   │   ├── nucleo-icons.ttf
│   │   ├── nucleo-icons.woff
│   │   └── nucleo-icons.woff2
```

```
│   ├── img
│   │   ├── apple-icon.png
│   │   ├── favicon.png
│   │   ├── logo.png
│   │   ├── profile-banner.jpg
│   │   └── profile.JPG
│   └── js
│       ├── core
│       │   ├── bootstrap.min.js
│       │   ├── jquery.min.js
│       │   └── popper.min.js
│       ├── paper-dashboard.js
│       ├── paper-dashboard.js.map
│       └── paper-dashboard.min.js
├── src
│   ├── dashboard.js
│   ├── transaction.js
│   └── user.js
├── dashboard.html
├── transaction.html
└── users.html
```

Tree-like formatted source code organization

2.3.2 Software Development Tools

Editor :

- **Visual Studio Code - Version 1.44.2 - https://code.visualstudio.com/**:

Visual Studio Code is a source-code editor developed by Microsoft for Windows, Linux and macOS. It includes embedded Git and support for debugging, syntax highlighting, intelligent code completion, snippets, and code refactoring.

Frontend :

- **Paper Dashboard 2 - https://www.creative-tim.com**

Paper Dashboard 2 is a beautiful Bootstrap 4 admin dashboard with a large number of components, designed to look neat and organized.

Backend Services:

- **Firebase - Version 6.22.0 - https://cocoapods.org/pods/Firebase**

Firebase is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. It is built on Google's infrastructure. Firebase is categorized as a NoSQL database program, which stores data in JSON-like documents.

2.3.3 Installation & Usage

You can clone the project using with following command:

```
git https://github.com/taylanakbas/Biometrix-Web-Admin.git
```

Go to project directory and open **dashboard.html**.

It automatically connects Firestore DB and Biometrix Data-API.